BABEŞ-BOLYAI UNIVERSITY CLUJ-NAPOCA

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

SPECIALIZATION APPLIED COMPUTATIONAL INTELLIGENCE

DISSERTATION THESIS

Fake news detection

Supervisor

Lect. Dr. Lupșa Dana

Author

**Teodora Ioana NASTE**

UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA

FACULTATEA DE MATEMATICǍ ŞI INFORMATICǍ

SPECIALIZAREA INTELIGENȚĂ COMPUTAȚIONALĂ APLICATĂ

LUCRARE DE DIZERTȚIE

Detectarea știrilor false

Conducător ştiinţific

Lect. Dr. Lupșa Dana

Absolvent

**Teodora Ioana NASTE**

Contents

[INTRODUCTION 4](#_Toc74419452)

[CHAPTER 1 4](#_Toc74419453)

[1.1 General information 4](#_Toc74419454)

[1.2 Data preprocessing 4](#_Toc74419455)

[1.3 Word embeddings 4](#_Toc74419456)

[CHAPTER 2 4](#_Toc74419457)

[2.1 Supervised learning 4](#_Toc74419458)

[2.2 Artificial neural networks 4](#_Toc74419459)

[2.2.1 Multilayer Perceptron 5](#_Toc74419460)

[2.2.2 Recurrent neural networks 5](#_Toc74419461)

[2.2.3 Long Short-Term Memory Neural Networks 6](#_Toc74419462)

[2.2.4 Gated Recurrent Neural Networks 7](#_Toc74419463)

[CHAPTER 3 8](#_Toc74419464)

[3.1 Stance detection vs. veracity detection 8](#_Toc74419465)

[3.1.1 Stance detection 8](#_Toc74419466)

[3.1.2 Veracity detection 10](#_Toc74419467)

[3.2 Fake news detection based on propagation graphs 11](#_Toc74419468)

[3.3 Satirical fake news detection 13](#_Toc74419469)

[CHAPTER 4 15](#_Toc74419470)

[CHAPTER 5 15](#_Toc74419471)

[CONCLUSION 15](#_Toc74419472)

[BIBLIOGRAPHY 16](#_Toc74419473)

# INTRODUCTION

Here comes some text for the introduction. Will be done last.

# CHAPTER 1

Fake News Detection

## 1.1 General information

## 1.2 Data preprocessing

## 1.3 Word embeddings

# CHAPTER 2

Machine learning models

## 2.1 Supervised learning

## 2.2 Artificial neural networks

The idea of a neural networks has first been introduced in 1943 by McCulloch and Pitts who have introduced the concept of a neuron as a conceptual unit capable of simple operations. However, as it is mentioned by Kröse, Ben et al. [6] , only in the 80s the interest towards artificial neural networks has begun to grow.

As the name suggests, the model of the artificial neural networks (ANN) is inspired by the structure of the human brain. It’s formed by a big number of artificial neurons, also called units or nodes, united by edges. Each unit *j*  has an associated activation function and each edge has a weight , which represents the cost from unit *j* to the unit *j’*. So, the value of a node, , is computed applying the **activation function**, like in the formula below:

The activation function is a simple linear function. The most common used ones are the sigmoid function: and the tanh function: . These are especially used for feedforward neural networks, but they have been applied to recurrent neural networks too. Another possible activation function is the rectified linear unit (ReLU) which is mostly used for deep neural networks and for the task of image detection.

The architecture of the neural network can be observed in Figure 2.2.1. On the leftmost side we have the input layer and on the rightmost side we have the output layer. The layers in the middle are called hidden layers. A neural network has only one input / output layer, but it can have however many hidden layers.



*Figure 2.2.1.Neural network architecture 1*

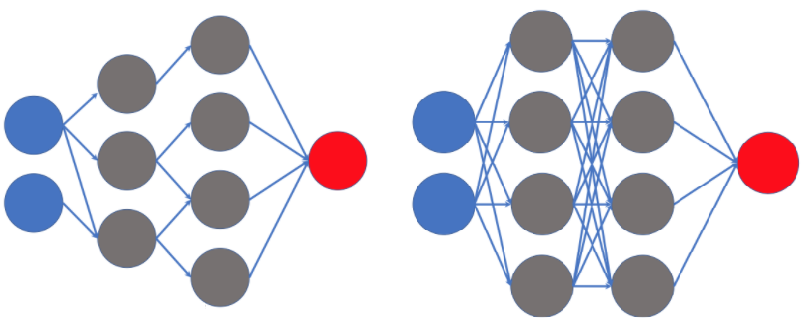
The artificial neural network is capable of approximating a non-linear function. For example, we have a function *f(x) = y* which mapsa value x with a value y. A neural network constructs a mapping function *y = f\*(x,θ),* where θ are the values of the parameters of the best approximation. The next question would be how the model chooses the best approximation and what exactly means best in this case. To answer, these networks learn using **error propagation**. The values calculated by the nodes are sent to the output nodes which compare them with the expected results. Each output node then returns the difference between the two , which is the error. The purpose is to minimize this error, so the best approximation is the one in which the error is at a minimum. The simplest method is to distribute the error from one output node to all the other nodes connected with it , proportionally with the nodes’ weights, till the input nodes and each edge will update its weight and a new error will be calculated. The error is calculated using a **cost function**. For that a smooth function like the quadratic cost or meansquare error is used to easily determined the small adjustmens of the weights to achieve the best approximation.

Some of the particular properties of artifical networks are their ability to to learn and adapt, to generalize a problem or to organize data. However, there are also some limitations to this model. Firstly, because we are talking about a superivised model, neural networks need a set of meaningful examples. Also, it can be hard to determinate how big the network should be, especially since for bigger networks (with many more layers) the training time can grow very much. Another thing that should be considered is that the training depends a lot on the processing power of the computer.

### 2.2.1 Multilayer Perceptron

The perceptron is an early type of neural network for binary classification with no hidden layers that consists of only one neuron. The concept was first proposed by Frank Rosenblatt [10] in 1958, based on the neuron proposed by McCulloch and Pitts. However, this type of neural network can only learn linear functions.

The multilayer perceptron is a special type of what was described earlier as a neural network. It uses multiple neurons and so, it has multiple (hidden) layers. The difference is that in an artificial neural network, the neurons from a layer are not necesarily connected to all neurons from the next or previous layer, while for amultilayer perceptrion the hidden layers are fully connected. Figure 2.2.2 better exemplifies the difference between the two types of neural networks.



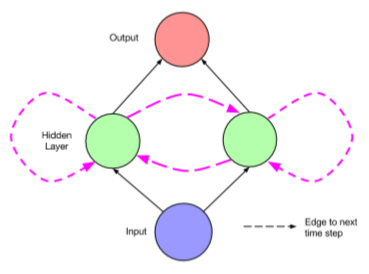
*Figure2.2.2. Left: graph representation of an artificial neural network*

*Right: graph representation of a multilayer perceptron*

A multilayer perceptron may also be defined as having the same number of neurons on the hidden layers or the same activation function across the layers. Also, they can refferenced as deep neural networks or feed-forward neural networks. The termn feed-forward comes from the fact that the information flow from inone way, from the input layer to the output layer (from left to right). There are neural networks in which information can be exchanged between the neurons of the same hidden layer too. These will be discussed in the next section.

### 2.2.2 Recurrent neural networks

In contrast to traditional neural networks, the recurrent ones can have cycles, introducing the notion of time to the model. Thus, at a time *t* , the nodes with a recurrent edge get information from the current input and from the previous states . A simple recurrent neural network (RNN) is presented in Figure 2.2.3. [7] Moreover, RNNs are able to handle variable-length input, making them suitable candidates for processing sequences like texts.



***Figure 2.2.3. Recurrent neural network [7]***

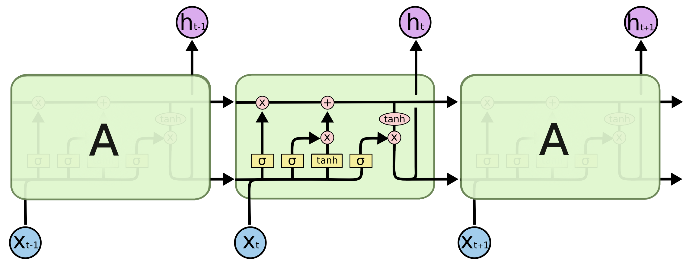
The training for a RNN can be difficult. The main problems that can appear are vanishing and exploding gradients which can occur during error propagation. Which of these two events may occur depends on the weight of the edges and on the activation function. For example, if the activation function is a sigmoid one, the values risk to get too close to 0 and the network won’t learn. In this case, we deal with vanishing gradient. If the activation function is a ReLU max(1,x), then it’s easy to imagine the values growing too high and the exploding gradient might appear.

### 2.2.3 Long Short-Term Memory Neural Networks

To solve the problems of the vanishing and exploding gradients that appear during training of a standard RNN, the Long Short-Term Memory neural network was proposed in 1997 [4] with an innovative gating system that introduces an intermediate memory space via the memory cell.

The gating system consists of 3 gates: the input, forget and output gates. The input gate, , decides how much of the information processed by the input node, , should be modified. The forget gate, helps to sort out the irrelevant information and the internal state, , is calculated based on the results from the previous 2 gates. Then, the final activation is composed with the help of the output gate, , which decides how much of the information should be passed to the next units. This process can be viewed in the next equations [7]:

Firstly, the hidden state computed by the previous nodes, , is concatenated with the input , then the result is passed through the forget gate, which uses a sigmod activation to ”*forget*” information that’s irrelevant for the current task. Next, the values go throught he input gate which also sues a sigmoid activation. At the same time, the candidate value is composed using a tanh activation. The candidates are then concatenated with results from the input gate. The next step is computing the internal state and the hidden statte that will be forwared to the rest of the nodes . This last step is done by the output gate. The internal state is normalized using a sigmoid function then multiplied with the candidate values . The whole flow can be also seen in Figure 2.2.4, which presents the structure of a LSTM neural network with its gates and activations.



***Figure 2.243. Structure of a LSTM neural network [7]***

LSTM neural networks have been successfully applied in many differen domains and are currently the most popular recurrent neural networks, even if they were firstly proposed over 20 years ago. Some of the fields in which they have been used are: text classification, text generation, automated translation, image captioning or predictions.

### 2.2.4 Gated Recurrent Neural Networks

The gated recurrent unit neural network is a type of recurrent neural networks proposed recently which use a gating system to solve the vanishing gradient problem similar to the LSTM. It was first proposed in 2014 by Kyunghyun Cho et al [2] as a simplified version of the aforementioned LSTM.

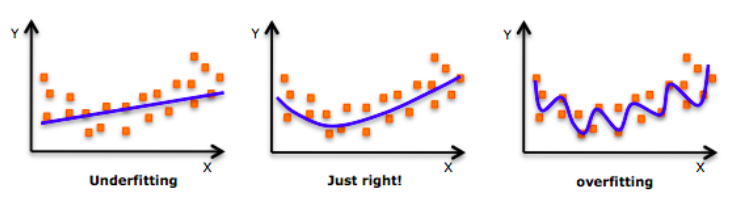
The gating mechanism is composed by 2 gates: the reset and the update gates. The reset gate eliminates the past information that is considered to be irrelevant, thus helping the current unit memory at moment *t*  to have only the important information stored. The update gate helps to decide how much of the past information to be passed along to the future. The final activation of the unit is composed using the previous hidden state and the results of the update gate, then passed to the next unit. Mathematically, this process consists of the next 4 equations [2]:

Each element of the input sequence is read by the model and the hidden state is computed based on the element and the previous hidden state. Then, the process is very similar to that of a LSTM in which the result from concatenating the input, however, the GRU doesn’t use an internal state anymore, but is counting on the hidden state to transport information throughout the network. While the update gate has a similar role like the forget and input gate, the reset gate is a bit different, because rather than deciding what to pass to the next unit, it decides what to forget from the past information.

## 2.3 Regularization

As mentioned in the previous chapters, recurrent neural networks are prone to overfitting. This is a common problem when training a neural network, however there are methods that help a model to avoid overfitting.

Firstly, what is overfitting? During training, it can happen that the model shows very good accuracy on the training data, but a poor one on the test data. This is because the model learnt too well the details in the edge cases from the training data. This can often happen especially when a model is too complex. Figure 2.3.1 shows an example of how overfitting looks compared to underfitting. It can be observed how the model fails to catch the nuances of the training data for underfitting and this leads to poor training accuracy. Comparatively, the mapping representation touches the edge points , creating a very specific curve. Because this curve is so specific to the training data, the model will fail on test data.



***Figure 2.3.1 Graphic representation of underfitting and overfitting***

While underfitting can be solved using a larger dataset or adjusting the parameters, overfitting can be avoided using regularization. Regularization is a technique that penalizes weights of the nodes in order to obtain small, but impactful changes for the model to generalize better.

There are different techniques used for regularization, but probably the most used ones are: L1 , L2 regularization and dropout.

**L1 and L2 regularization** are probably the most common ones. As their name suggests they are related to the L1 and L2 norms of a vector *w*. In this context, these will be the regularization terms.

The regularization terms are added when computing the cost function. Assuming *y* is the function we want to approximate, *y’* is the computed approximation and λ is the regularization parameter who is optimized for better results, the loss would look like this:

Adding the regularization terms to the loss, we force the weights to shrink to values close to 0 (even 0 in case of L1). This is done because it is assumed that neural networks with smaller weights lead to simpler models. Now, to explain why adding these terms help in dealing with overfitting. Let’s assume that *y’* generated a perfect match for an edge case in our model. This leads to overfitting, but with the help of the regularization, the value is slightly shifted from the noise data. Also, we don’t want perfect matches in our model, because, without regularization, this will lead to no real change in the weights and so the model won’t be able to predict well on other datasets, so adding the regularization also helps with the generalization.

**Dropout** is a different type of regularization, but is also the one that produces the best results and is the most frequently used in deep learning, however still very simple and approachable. The idea behind this technique lays in the concept of ensembles. Ensembles are a group of models which can better cover different aspects of the data and so they can be used to reduce overfitting and enhance generalization. However, there is a cost in computational power and maintenance for multiple models. So, through dropout an ensemble can be simulated using only one single model. This is done by randomly dropping some nodes and their edges during each training iteration. So, with each iteration we can get a different neural network structure and ,in effect, each layer will update the weights differently for the nodes. In this way it adds more randomness to the model, encouraging a better generalization This technique is especially good when dealing with large and complex neural networks.

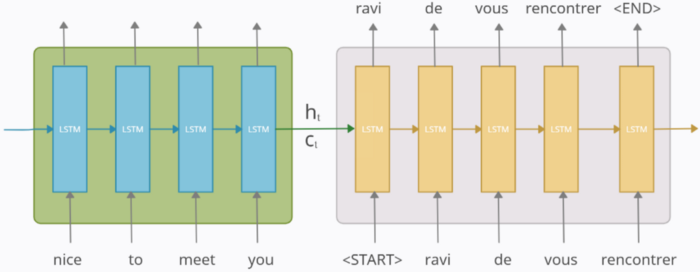
Dropout is implemented on each layer and it can be used on any types of layers (hidden, input/output). A hyperparameter is also given that indicates the probability of dropping, or inversely, keeping a node, also known as dropout rate. This probability is usually 0.5 for a hidden layer and somewhere close to 1 for a visible layer (input or output) [11]. Of course, because of the dropout , the weights would be larger than normal, so before finishing, they should be scaled down with the chose dropout rate.

## 2.4 Transformer models

Transformers are a quite recent addition to machine learning and especially to the field of natural language processing. This field has been dominated for quite a while by recurrent neural networks based models, like Long Short-Term Memory and Gated Recurrent Unit, because of their powerful ability to process sequences and remember important details. However, transformers offered a breakthrough , proving to surpass the old models for certain tasks like machine translation [8],without using any recurrent neural networks at all. The architecture of the transformer was first proposed by Vaswani Ashish et al. in 2017 and the idea was based on 2 other concepts: the sequence-to-sequence models and the attention mechanism, so before going further into describing the transformer model, I’ll first explain these concepts.

### 2.4.1 Sequence-to-sequence (Seq2Seq) model

A Seq2Seq model is actually a neural network that is given a sequence of elements, such as a sentence, transforms it into another sequence, as the name also suggests. These models are based on the Encoder-Decoder architectures such as the one proposed by Cho et al. [2] and Sutskever Ilya et al. [13]. Both of these papers focus on machine translation and express their motivation for such a system because the neural networks, even though are powerful tools in learning difficult tasks with large labeled data, they still lack in mapping a sequence to another sequence. These models are also achieving spectacular results for other tasks like image captioning, question answering etc. In fact, Google has started using a seq2seq model for translation since 2016. The architecture of the Seq2Seq can be observed in Figure 2.4.1 [9].



***Figure 2.4.1 Architecture of the Encoder-Decoder model***

As mentioned, the model is formed by 2 parts: an encoder and a decoder. These are usually some type of a recurrent neural network, like GRU [2] or LSTM [13]. The encoder is given an input sequence and it translates it into a context vector. The purpose of the context vector is to encapsulate the important information that need to be then decoded to make accurate predictions. For this step the actual outputs of the encoder are ignored and only the context vector or internal states are considered. Then, the decoder gets the final state of the encoder and it translates into the desired sequence output. So, while the encoder is trained on the input sequence, the decoder is actually trained on the information received from the encoder. To better explain how the decoder works, we should look at the Figure 2.4.1, at the decoder side. The first input is a START symbol which is expected to generate the first word in the translation. Then the output is passed as the next input until an END symbol.

There are however disadvantages to a simple seq2seq model and these are mostly related to the fixed length of the context vector. Due to this, for long sentences, the context might forget important information from the beginning of the sentence till it reaches the end.

### 2.4.2 Attention mechanism

The attention mechanism offers a way for a neural network to distinguish between what is relevant from the input and what is not. This is extremely useful because given a sentence, not all words from the sentence are actually important to the meaning, but in fact it can be reduced to a few important words. Because of this feature, attention can help where the context vector’s fixed length is lacking.

Now, if we take the concept of seq2seq models, we can see how to incorporate attention to give the decoder more insight depth in the data. On the encoder part, for each input, the attention mechanism will compare it with several other inputs and assign weights that would quantify the importance of the input (words). These will then be sent to the decoder which can use the weights as extra information for better predictions.

### 2.4.3 Transformers

# CHAPTER 3

Applications

The raise in popularity of social media has massively increased the amount of user-generated information that can spread uncontrollably throughout the web. This huge amount of data generated in real-time is impossible to be filtered and checked manually. So, there has been a growing interest to research automatic ways of detecting false information. However, considering the complexity of what can be considered fake news and the various forms in which this can be spread in, the task of simply determining the authenticity of the information can be very hard and complex. In this chapter, I will present some interesting and diverse methods that have been applied for this task.

## 3.1 Stance detection vs. veracity detection

When talking about fake news detection, most people would think of labelling a claim, article, headline or post as either true or false. This is, more or less, what veracity detection aims to do. However, in practice, this is more complex that it sounds. First of all, research in the area has been done heavily only in the past 4-5 years. This leads to data being scarce and unstructured. Stance detection came as a response to these shortages, providing a first step towards fully automated fact checking. The idea behind it is to possibly help the professional fact checkers by offering relevant background information about the text based on the stances of various organizations on the topic, weighted by their credibility.

Both approaches have the same overall goal, but they are different in ways each is approach, the data they need or the inputs and outputs of the models. Also, the solutions from one task can inspire research for the other task and that’s why I think it is relevant to describe some of these methods.

### 3.1.1 Stance detection

Realising the complexity that fake news detection consists of, as a first step to tackle the task, in 2017 the Fake News Challenge[[1]](#footnote-1) released a contest for stance detection. Given two pieces of text, the purpose of the task was to estimate the relative perspective or stance of a text to the other on a given topic, claim or issue. The dataset for the challenge consisted in pairs of news article headline and its body and the contestants had to estimate the stance of the body towards the headline. The stance was represented by one of the following labels: *agree, disagree, discuss* orunrelated. There were many entries for this challenge, each with different types of models, however the ones that got into top 3 used rather simple methods.

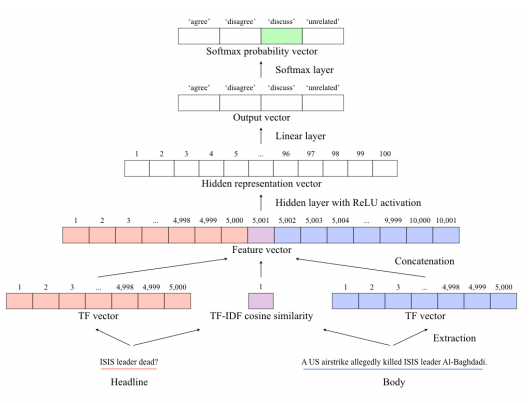
One of the approaches that got into top 3 was proposed by Benjamin Riedel et al. [1]. They probably had the most simple method, yet managed to get a considerable high accuracy.

**Methodology**

The method proposed consists of passing lexical and similarity features through a multi-layer perceptron with one hidden layer. The text was represented by a simple bag-of-words and the features considered were the following:

* Term frequency (TF) vector of the headline
* Term frequency (TF) vector of the body
* Cosine similarity between the TF-IDF vectors of the body and headline

The MLP classifier had 100 units on its single hidden layer. The rectified linear unit (ReLU) was used as the activation function and a softmax function on the output layer. In figure 3.1.1 a schema of the system is presented.

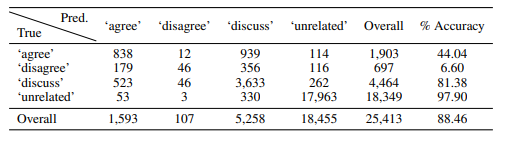


**Figure 3.1.1 Schematic diagram of the stance detection system**

A dropout and the Adam optimiser were also applied to avoid the gradient descent issue. The parameters for the model were learnt using a random search and cross-validation.

**Results**

The proposed method reached an accuracy of 81.72% which took the team on the third position at the end of the challenge. Results on different labels are detailed in Table 3.1.1., where it can be observed that the system performed best on text labelled with “agree”.



**Table 3.1.1 Confusion matrix for the stance detection system**

### 3.1.2 Veracity detection

When talking about fake news detection, the first concept that pops into anyone’s mind is to check the information (content) from a piece of news and assign if it is true or false, without necessarily taking into account the intention of the creator for example.

Such approach has been taken by Souvick Ghosh an Chirag Shah [12] in their research paper *Toward automatic fake news classification*. Inspired by one of the top 3 teams from Fake News Challenge 2017 they wanted to see if combining information retrieval with stance detection and deep learning, they will be able to obtain good results on labelling text as ‘Fake’, ‘Legit’ or ‘Suspicous’.

**Dataset**

Because of a lack of benchmark datasets , the researchers used multiple datasets which they combined into 2 categories:

* Type 1, formed from LIAR, Kaggle’s fake News dataset and Fake News Challenge dataset, is a set of short texts with the length between 70 and 150 characters, including statements and article headlines.
* Type 2, formed from University of Washington Fake News Dataset, consists of much larger texts with the length between 400 to 700 words.

**Methodology**

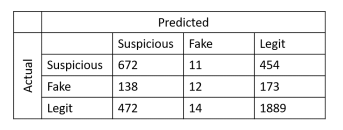
A submodular approach has been used. The first submodule was assigned to detect the veracity of the text using information retrieval, after previously constructing a knowledge base. First, documents referring the claim are retrieved from the knowledge base. For this the TF-IDF method was used, combined with a other advanced algorithms (BM25, Vector Space model, Language model) to improve the performance. After that, for each document retrieved the stance of the claim towards the document is determined. The second step has the task of classifying the claims into one of the 3 categories mentioned above. For that , they used a simple Feed Forward Neural Network that gets as input the bags-of-words vectors for the claim and the document, as well as the cosine similarity between these vectors. For training, the Type 1 dataset was used, as it is more focused on fact-based differences.

The second submodule is concerned with finding patterns in the writing style of the claim. It has been observed that fake news tend to be more aggressive and depict stronger emotions in order to manipulate the audience [14]. For this task, a bidirectional LSTM has been used, as these networks are known for their capabilities to process long sentences. This submodule was trained using the Type 2 dataset, as it has more richness in style compared to the short texts from Type 1.

Finally, results from both submodules are combined based on a voting system with the help of a weighted average.

**Results**

The prediction submodule managed to obtain an accuracy 67.1% for the ternary classification and of 72,12% for the binary classification, while the style based submodule obtained an accuracy of 81.83%. After combining the two submodules, they obtained a total accuracy of 82.4%. The experimental results for the prediction on FNC Dataset can be seen in Table 3.1.2.

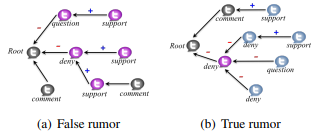


**Table 3.1.2 Confusion matrix for FNC Dataset**

## 3.2 Fake news detection based on propagation graphs

Social media is the main environment in which fake news are being created and spread at an alarming rate. However, this makes space to research different aspects of fake news, like the propagation pattern and the users’ response via comments or reposts, with the intention to have an automatic method of spotting misinformation in its early stages of spreading. Veracity detection has been used for detecting fake news in social media, but the difference here is manner in which the true/false tags are assigned: instead of checking the information from a post, it looks at the way it propagates.

In [5], have proposed a tree-structured recursive neural network approach that analyses how a rumour is spread and what’s the users’ stance relative to it. They started this researched based on observations they have made that suggest that if a user denies a post with a fake rumour, it tends to trigger positive responses from other users, confirming the denial, while denying a true rumour triggers negative and confusing responses. This pattern can be seen in Figure 3.2.1.



**Figure 3.2.1 Propagation tree containing users’ stance towards the post**

**Methodology**

A rumour (claim) was defined as an original tweet together with all its relevant responsive tweets ordered chronologically. For these claims the task is to be classified into one of the following categories:  *non-rumour, false rumour, true rumour* and *unverified rumour.* To capture the information from different angles, two variants have been adopted:

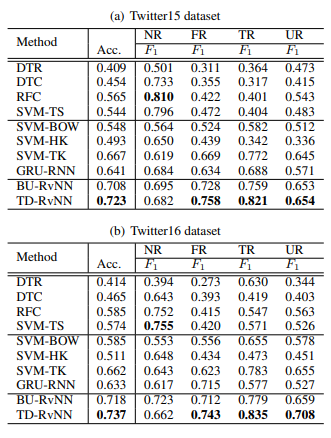
* Bottom-up RNN in which the direction of the edge points to the source, like in Figure 3.3.1
* Top-Down RNN which follows the natural propagation course in which the direction of an edge goes from the source to the user node that responded to that post

The purpose for each tree is to generate a feature vector that can be used as input to an extended version of the GRU neural network that Jing Ma et al. implemented. The difference between the two variants however is that, the bottom-up is going to start from the leaves and create a feature vector for each subtree and merging them together at the root creating the vector for the whole tree. On the top-down version, the creation is starting from the original post , resulting in feature vectors for each propagation line, since the final representation will be in the leaves of the tree. This enables a deeper insight into the complex propagation patterns that rumours can have on social media.

The error function used was the squared error. The parameters were updated using back-propagation through structure and also a gradient-based optimization was applied.

**Results**

The evaluation was done on two real-life Twitter datasets (Twitter15 and Twitter16) which each contain over 1000 propagations trees. To better present the advances of their work, Jing Ma et al. made an elaborate comparison with multiple state-of-the-art methods. The results on both sets can be seen in Table 3.2.1



**Table 3.2.1 Results comparison of rumour detection for each category**

From the table, it can be easily observed how the proposed model surpasses previous ones on most categories. Moreover, after obtaining such amazing results, further experiments were done to see the capabilities to detect rumours in early stages. They concluded that their method needs approximately 8 hours or about 90 tweets to correctly spot a fake rumours which is superior to the best baseline method which needs 36 hours and about 300 posts.

## 3.3 Satirical fake news detection

Satirical news bring another challenge to fake news detection, because their purpose is for entertainment and thus, it is harder to recognize the intention of the creator. Also, the satire can be subtle and so, users could easily misunderstood it as true news. The style of writing these kind of news it’s different than with other types of articles and it needs special investigation. Therefore, satirical news detection is researched separately from other news detection.

Considering this, Fan Yang et al. [3] have investigated satirical news and proposed a paragraph-level detection model that can deal with such subtle satire, but also highlighting the difference in features between document-level and paragraph-level. As they also pointed, satirical news are still considered fake news if the user believes them and spreads them as true information.

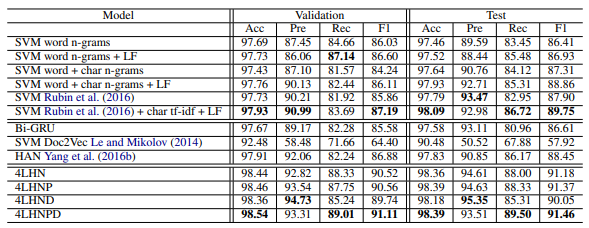
**Methodology**

To achieve their goal, they proposed a 4-level hierarchical neural network, following the hierarchy of character-word-paragraph-document.

Firstly, on the lowest level of the hierarchy, a convolutional neural network (CNN) is used to encode words representations from characters. Then these representations are sent one level up, where based the word embedding takes place. For encoding a Gated recurrent Unit (GRU) was implemented. The third level focuses on encoding at paragraph level, using a bidirectional GRU. Here, the attention mechanism is included because it was observed that not all paragraphs contain satirical cues and some paragraphs can be more relevant than others. For that, a satirical degree is being computed based on the paragraph representations and the results from the hidden states. The final document is the weighted sum of the satirical degree and the paragraph representations. On this level extra linguistics features are added. On the final level the final vector representation is computed via MLP and the prediction is done using a sigmoid activation.

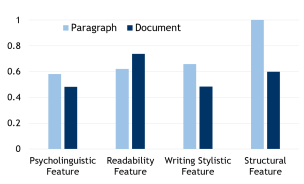
**Results**

For evaluation, over 16.000 satirical news were collected from various sites that focus on such content and over 160.000 true news from major news outlets. Since the data is not annotated, only the binary classification was taken into account. Moreover, different models were implemented to prove the efficiency of the method. These models as well as all the results can be seen in Table 3.3.1.



**Table 3.3.1 Results comparison for satirical news detection**

It can be observed that the model including both paragraph and document level features performs the best in almost all categories. The importance of the linguistic features used was also analysed which gives a very interesting insight into what features are more relevant when trying to detect satire. The results can be seen in Figure 3.3.1.



**Figure 3.3.1 Comparing the importance of the four features sets**

# CHAPTER 4

Implemented solution

# CHAPTER 5

Future work and improvements

# CONCLUSION

# BIBLIOGRAPHY

[1] Benjamin Riedel et al. , *A simple but tough-to-beat baseline for the Fake News Challenge stance detection task, 2017*

[2] Cho, Kyunghyun et al. , *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation ,* 2014

[3] Fan Yang et al., *Satirical news detection and analysis using Attention Mechanism and Linguistic Features,* 2017

[4] Hochreiter, S., & Schmidhuber, J.. *Long Short-Term Memory. Neural Computation, 9(8), 1735–1780.* doi:10.1162/neco.1997.9.8.1735 , 1997

[5] Jing Ma, Wei Gao and Kam-Fai Wong, *Rumour Detection on Twitter with Tree-structured Recursive neural Networks*, 2018

[6] Kröse, Ben, et al. , *An introduction to neural networks*, 1993

[7] Lipton, Zachary C., John Berkowitz, and Charles Elkan. *A critical review of recurrent neural networks for sequence learning*. arXiv preprint arXiv:1506.00019 , 2015.

[8] Maxime, *What is a transformer?*, on <https://medium.com/inside-machine-learning>, 2019

[9] Moses Kriz, *Encoder-Decoder Seq2SeqModel, Clearly Exmplained,* on <https://medium.com/analytics-vidhya/>, 2021

[10] Rosenblatt Frank, *The perceptron: a probabilistic model for information storage and organization in the brain.* Psychological review 65.6 , 1958 : 386

[11] StrivastavaNitish et al., *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*, The journal of machine learning research 15.1, 1929-1958, 2014

[12] Souvick Ghosh and Chirag Shah*, Toward Automatic Fake News Classification, 2017*

[13] Sutskever Ilya et al., *Sequence toSequence Learning with Neural Networks*,Advances in neural information proessing systems , 2014

[14] Xinyi Zhou and Reza Zafarani, *A Survey of Fake News: Fundamental theories, detection methods and opportunities,* 2020

[a] Gong Steven, *How does a Neural Network work intuitively in code?* , on <https://gongster.medium.com/>, 2019

[b] Jain Shubham, *An overview of Regularization Techniques in Deep Learning,* on <https://www.analyticsvidhya.com/>, 2018

1. <http://www.fakenewschallenge.org/> [↑](#footnote-ref-1)