Combining Similarity Features and Deep Representation Learning for Stance Detection in the Context of Checking Fake News

Luís Pedro Pires Borges

Thesis to obtain the Master of Science Degree in Information Systems and Computer Engineering

Supervisors: Prof. Bruno Emanuel da Graça Martins
                      Prof. Pável Pereira Calado

Examination Committee
Chairperson: Prof. Francisco João Duarte Cordeiro Correia dos Santos
             Supervisor: Prof. Bruno Emanuel Da Graça Martins
               Member of the Committee: Prof. Diogo Manuel Ribeiro Ferreira

October 2018
Acknowledgments

First of all, I would like to thank my supervisors, Prof. Bruno Martins and Prof. Pável Calado, for their continuous support throughout the development of my M.Sc. research project. They were always available to share their knowledge, to read preliminary versions of my dissertation, and to give advice on how I could improve the results of my research.

I also express my gratitude to INESC-ID for funding my research project, through a grant aiming to support M.Sc. students, and for providing me and my research colleagues a room where we could gather and work together on our projects. Those same colleagues were always very helpful during the development of this work, and my appreciation goes out to them as well. Furthermore, a special thanks goes to Unbabel, whose reading group I was privileged to be included in, and also to Priberam, who offered to fund my participation in the 2018 edition of the Lisbon Machine Learning School (LxMLS), and where I did a brief summer internship.

To all the friends and colleagues who were there for me, either for work related issues or for some much needed distractions, my sincere thank you. My five year journey at Instituto Superior Técnico would not have been the same without you.

Finally, I would like to thank my family, who always gave me the moral and material support I needed while studying in Lisbon – I cannot thank them enough and I love them very much.
Resumo

Neste trabalho, são abordadas técnicas para auxiliar a detecção de notícias falsas. Concretamente, esta dissertação propõe uma abordagem para resolver o problema da detecção automática de postura (stance detection), tendo como base o problema de inferência em língua natural. Em ambas as tarefas, o desafio é prever o tipo de relação entre dois pedaços de texto. No caso da tarefa de detecção de postura, esta previsão é feita através de uma rede neuronal que combina word embeddings, unidades recorrentes, atenção neuronal, max-pooling, e extração de características externas. Esta rede neuronal explora também a natureza hierárquica dos dados à disposição, modelando cada frase como uma sequência de palavras, e um documento como uma sequência de frases.

Uma das contribuições principais desta tese assenta na quantificação da utilidade da tarefa de inferência entre frases para a tarefa de detecção de postura. Tal hipótese é demonstrada inicializando os pesos da componente responsável por codificar frases na arquitetura neuronal da tarefa de detecção de postura, com os pesos da componente homóloga da tarefa de inferência. Para além disso, foram também realizados testes onde os modelos de inferência são utilizados na tarefa de detecção de postura para criar um alinhamento entre o título de um artigo e as várias frases que o compõem.

Os resultados experimentais confirmam o contributo das componentes acima descritas. O melhor modelo atinge valores de exatidão de 83.38% na tarefa de detecção de postura proposta no contexto do Fake News Challenge, definindo um novo estado-da-arte nessa mesma tarefa.

Palavras-chave: Deteção de Notícias Falsas, Deteção de Postura, Inferência em Língua Natural, Aprendizagem com Redes Profundas, Processamento de Língua Natural, Inteligência Artificial
Abstract

This work addresses the development of techniques to assist on the detection of fake news. Concretely, this dissertation proposes an automated approach to address the stance detection problem, based on the natural language inference task. In both tasks, the challenge is to predict the kind of relation between two pieces of text. In the case of stance detection, this prediction is obtained by leveraging a neural network that combines word embeddings, recurrent units, neural attention, max-pooling, and feature engineering. This neural network also explores the hierarchical nature of the available data, modelling sentences as sequences of words, and each document as a sequence of sentences.

One of the main contributions of this thesis lies in quantifying the usefulness of the natural language inference task for the stance detection problem. This hypothesis is demonstrated when initializing the weights of the sentence encoding component in the stance detection architecture with the weights of the equivalent component in the natural language inference task. Furthermore, several tests were conducted using the natural language inference models in the stance detection problem, to generate an alignment between the headline of an article and several sentences that compose it.

The experimental results attest to the usefulness of the contributions mentioned in the previous paragraphs. The best model achieves a weighted accuracy score of 83.38% in the stance detection task proposed in the context of the Fake News Challenge, hence defining a new state-of-the-art result.

Keywords: Fake News Detection, Stance Detection, Natural Language Inference, Deep Learning, Natural Language Processing, Artificial Intelligence
# Contents

Acknowledgments ................................................................. iii
Resumo ................................................................. v
Abstract ................................................................. vii
List of Tables ............................................................... xi
List of Figures ............................................................ xiii
Glossary ................................................................. 1

1 Introduction ................................................................. 1
  1.1 Objectives ............................................................ 2
  1.2 Methodology .......................................................... 2
  1.3 Results and Contributions ........................................ 3
  1.4 Thesis Outline ...................................................... 4

2 Concepts and Related Work ........................................... 7
  2.1 Representing Textual Information .................................. 7
  2.2 Introduction to Neural Networks ..................................... 9
    2.2.1 The Perceptron .................................................. 9
    2.2.2 The Multi-Layer Perceptron ................................... 10
    2.2.3 Convolutional Neural Networks ................................ 10
    2.2.4 Recurrent Neural Networks .................................... 11
  2.3 Deep Learning Methods for Natural Language Processing ..... 12
    2.3.1 Text Classification ............................................. 13
    2.3.2 Natural Language Inference .................................... 15
  2.4 Stance Detection in the Context of the Fake News Challenge 19
  2.5 Overview ............................................................ 24

3 The Deep Neural Model for Fake News Detection .................. 27
  3.1 The Sentence Encoder ............................................... 27
  3.2 The Hierarchical Approach for Encoding Documents .......... 29
  3.3 Model Pre-Training with the SNLI and MultiNLI Datasets ..... 29
  3.4 Combining the Representations with External Features ...... 30
  3.5 Summary ............................................................ 32
List of Tables

2.1 Accuracy of previous models for the natural language inference task, over the SNLI and MultiNLI testing datasets. ........................................ 24
2.2 Results obtained with the testing split of the FNC-1 dataset. ......................... 25
4.1 Characterization statistics for the FNC-1 dataset. ........................................ 36
4.2 Characterization statistics for the SNLI and MultiNLI datasets. ....................... 37
4.3 Accuracy on the natural language inference task, over the SNLI and MultiNLI testing datasets. ...................................................... 38
4.4 Results obtained over the testing split of the FNC-1 dataset. ......................... 39
4.5 Experimental results with different variations of the proposed method, over the testing split of the FNC-1 dataset. ................................. 40
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Overview of a generic NLI approach, adapted from the description by Conneau et al. [7].</td>
<td>16</td>
</tr>
<tr>
<td>3.1</td>
<td>Overview of the proposed approach.</td>
<td>28</td>
</tr>
<tr>
<td>3.2</td>
<td>A sentence encoder based on bi-RNNs, neural attention and max-pooling for processing an input sentence.</td>
<td>29</td>
</tr>
<tr>
<td>4.1</td>
<td>Attention weights computed with the complete model, for two example instances from the FNC-1 test split.</td>
<td>41</td>
</tr>
<tr>
<td>4.2</td>
<td>Distribution for the neural attention weights associated to two distinct refutation words, within FCN-1 testing instances belonging to the disagrees versus the agrees class.</td>
<td>42</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Fake news articles (i.e., made up stories with the intention of deceiving, and that, most of the times, are used to achieve secondary gains) are undoubtedly one of the most serious challenges that journalists and the news industry are facing today. Given the ease in obtaining and spreading information through social networking platforms, it is increasingly harder to know for sure what to trust, with the absorption of fake news by the masses having increasingly harmful consequences [1]. Automatically dealing with fake news has drawn considerable attention in several research communities [2]. However, the task of evaluating the veracity of news articles is still very demanding and complex, even for trained specialists and much harder still to automated systems.

A useful first step towards identifying fake news articles relates to understanding what other news agencies, in a given moment, are reporting about the same topic. This sub-task is often referred to as stance detection, and automating this process might be useful in developing automated assistants to help in fact checking. In particular, an automatic approach to stance detection would allow, for example, someone to insert an allegation or a news title, and recover related articles that either agree, disagree, or discuss that title. Then, the human checker would use his own judgment to assess the situation.

Based on the aforementioned general ideas, a Fake News Challenge (FNC-1) was organized in early 2017 by a group of academics and contributors from the journalism industry, to foster the development of systems applying artificial intelligence and machine learning for evaluating what a news source is saying about a particular issue. Specifically, FNC-1 involved developing models that, given the body of a news article and a news headline, estimate the stance of the article towards the headline (i.e., the article can either agree, disagree, discuss, or be unrelated to the headline). More information on the FNC-1 task, its rules, the dataset, and the evaluation metrics, can be found on the official website¹. A total of 50 teams actively participated on the challenge, and the training/testing data splits have been released afterwards, in order to encourage further developments.

¹http://fakenewschallenge.org
1.1 Objectives

This dissertation presents a method for the stance detection task associated to detecting fake news, where the goal is to classify the relation between a headline and a news article.

The main objective was to study how the data from the FNC-1 can be classified using a supervised machine learning approach, specifically a deep neural network, leveraging state-of-the-art methods proposed not only for stance detection but also for other related challenges, such as natural language inference, or text classification.

Despite the good results achieved by the participants in FNC-1 (i.e., scores of 82.02% and 81.98% for the two winning teams, in terms of the accuracy metric considered for the challenge), these same results showed that standard approaches based on convolutional and/or recurrent neural networks are not well-suited to this task, as they fail to model semantic relations with respect to large pieces of text (i.e., the bodies of news articles, composed of multiple sentences). This work also aimed to verify if the combination of feature engineering with deep learning approaches could achieve state-of-the-art results for the aforementioned task.

Finally, a last objective was to quantify how beneficial the natural language inference problem can be for solving stance detection through model pre-training, e.g. by initializing the weights of the FNC-1 neural architecture with the weights from an analogous NLI approach.

1.2 Methodology

In the first stage of this work, a general revision was conducted on the area of Natural Language Processing (NLP), not only concerning the stance detection and FNC-1 tasks, but also other NLP challenges such as Natural Language Inference (NLI) and text classification. Relying on the conceptual similarity between natural language inference and stance detection, the conclusion that a strong approach for NLI would be a good starting point for stance detection arised. Therefore, a more extensive analysis was conducted on the state-of-the-art for the NLI task, and several experiments leveraging deep learning approaches began to be made for NLI.

After establishing a good baseline model for NLI, the next step was to transfer this approach into the main task of this dissertation, i.e. the FNC-1 stance detection problem. Taking inspiration on several participations in the challenge, a feature engineering based component was introduced, which constituted a major breakthrough in the development of this work. Finally, efforts were dedicated to establishing a more solid connection between NLI and the FNC-1, e.g. when leveraging NLI approaches to create an alignment between the headline of the article and the first sentences that compose it. This dissertation contains ablation studies analyzing how each of these components affect the final results.

Regarding the technologies that were considered when implementing this project, the Python programming language was chosen, given its popularity and vast documentation. More concretely, this research project used a Python deep learning library named Keras\(^2\), which in turn uses Theano\(^3\) or

\(^{2}\text{http://keras.io/}\)
\(^{3}\text{http://www.deeplearning.net/software/theano/}\)
TensorFlow\(^4\) as the computational backend.

In order to train the proposed neural networks, the SNLI\(^5\) and MultiNLI\(^6\) data was considered for the NLI task, and the official FNC-1 data was considered for the stance detection task. The data partitioning into training/testing datasets was the same as proposed by the authors of the datasets, i.e., no custom splits were made. The considered metrics for evaluation were the accuracy, when dealing with the SNLI and MultiNLI corpora, and a special weighted accuracy for the Fake News Challenge task, which gives extra weight to some classes which are less represented in the training and testing datasets. Chapter 4 goes into further detail regarding the experimental evaluation employed in this dissertation.

1.3 Results and Contributions

The main ideas and contributions of this work can be summarized as follows:

- A hierarchical method for modeling the body of news articles is proposed, taking inspiration on previous studies addressing the classification of long documents [3, 4]. In this approach, a Recurrent Neural Network (RNN) is used for modeling the sequence of sentences, which in turn are individually modeled by a separate RNN encoding sequences of words. Specifically, either bi-directional Long Short-Term Memory (LSTM) units or bi-directional Gated Recurrent Units (GRUs) can be leveraged as the encoders, together with a max-pooling operation and/or a neural attention mechanism that weights the individual word representations, for encoding text in the proposed model (i.e., the headline, and the sentences in the news article being matched). Separate layers of bi-directional GRUs/LSTMs, also combined with max-pooling and/or neural attention, are used for encoding the sentences that constitute the document. This approach can directly incorporate knowledge of the document structure in the model architecture, at the same time also exploring the intuition that not all sentences/words will be equally relevant for predicting the stance;

- Taking inspiration on previous studies addressing natural language inference [5, 6], a deep neural network architecture is proposed for stance detection, as defined in the FNC-1 task. The proposed architecture leverages pre-trained word embeddings, uses shared parameters for some of the components (e.g., the sentence encoders), and attempts to match multiple representations learned from the inputs. Specifically, representations for three different inputs are combined/matched (i.e., the headline, the first two sentences of the news article, and the entire document) through a series of operations known to perform well on natural language inference [7], namely the vector concatenation, difference, and element-wise product. A final layer processes this result, and predicts the stance of the news article towards the headline.

- The SNLI [5] and MultiNLI [6] datasets of sentence pairs, previously proposed for evaluating natural language inference models (i.e., models for checking if a given hypothesis sentence entails, contradicts, or is neutral towards a given premise sentence), were used to pre-train some of the

\(^4\)http://www.tensorflow.org/

\(^5\)http://nlp.stanford.edu/projects/snli/

\(^6\)http://www.nyu.edu/projects/bowman/multinli/
components involved in the proposed neural network architecture. Previous studies have already attested to the benefits of similar pre-training procedures in other natural language processing tasks [7]. Specifically, the SNLI/MultiNLI sentence pairs were used to pre-train the components involved in encoding sentences, and also the part of the network that matches the headline with the first two sentences of the document.

- Learning effective representations for news article bodies can be quite challenging for neural methods, although this is essential for achieving good performance on the FNC-1 task [8, 9]. Therefore, this work proposes to combine the representations learned through neural network layers, with external similarity features. Specifically, features from previous FNC-1 methods were used (e.g., from the baseline method introduced by the organizers of the challenge), together with other methods proposed for similar text matching problems (e.g., BLEU and ROUGE scores [10, 11], the soft-cosine similarity metric [12], or the CIDEr score [13]).

- An extensive set of experiments was conducted, evaluating the contribution of each of the aforementioned methods. The results confirm that model pre-training can indeed improve the overall accuracy. Moreover, despite good results associated with the use of the hierarchical attention method for encoding the news articles, the results also show that external similarity features are highly beneficial. The complete method establishes a new state-of-the-art result for the FNC-1 dataset, outperforming the previous approach described by Bhatt et al. [8].

The best model achieved a state-of-the-art weighted accuracy of 83.38% on the FNC-1 testing dataset. Regarding the NLI datasets, the approach used for model pre-training scored an accuracy of 84.1%, 70.8%, and 70.8%, respectively when considering the SNLI, MultiNLI matched, and MultiNLI mismatched testing datasets.

During the development of this research project, an article was produced entitled Combining Similarity Features and Deep Representation Learning for Stance Detection in the Context of Fake News. This paper was submitted to a special issue of the ACM Journal of Data Information and Quality (JDIQ), and is currently awaiting an acceptance decision (i.e., a first round of interaction with external evaluators suggested a few minor revisions).

The source code corresponding to the implementation of the different approaches proposed in this dissertation can be found on GitHub7.

1.4 Thesis Outline

The rest of this dissertation is organized as follows:

- Chapter 2 surveys important concepts and related work. First, Section 2.1 explains several ways of representing textual information in vector form. Then, an overview of machine learning methods leveraging deep neural networks is presented in Section 2.2. Section 2.3 reviews how deep

7http://github.com/LuisPB7/fnc-msc
learning methods can be applied in Natural Language Processing tasks such as text classification and natural language inference. Finally, a description of previous work related to the Fake News Challenge (FNC-1) is provided in Section 2.4.

• Chapter 3 details the proposed approach for the stance detection task at hand, presenting the neural network architecture that was considered to address the Fake News Challenge (FNC-1).

• Chapter 4 presents the experimental evaluation of the proposed approach. The chapter starts by detailing the datasets used in the experiments, together with the experimental methodology and the custom evaluation metric for FNC-1. Furthermore, the chapter presents an ablation study on how the components from the proposed architecture influence the final results. Finally, the chapter discusses how the weights in the neural attention layer can be used for interpretation purposes.

• Lastly, Chapter 5 overviews the main conclusions regarding this project, and points to several directions for future work.
Chapter 2

Concepts and Related Work

This chapter presents fundamental concepts and previous work that addressed not only the fake news detection task at hand, but also other relevant NLP problems such as natural language inference and text classification. Section 2.1 overviews popular methods for representing documents and words in vector form, useful for supervised learning. Section 2.2 presents an introduction to several types of neural networks. Section 2.3 reviews previous work on the NLP tasks of text classification and natural language inference, while Section 2.4 introduces the Fake News Challenge and also describes previous work on this task. Finally, Section 2.5 summarizes the related work presented in this chapter.

2.1 Representing Textual Information

This work addresses the application of supervised machine learning techniques to a natural language processing task. In brief, supervised learning concerns with inferring the parameters of models that take as input a fixed dimension vector $x$ and return as output another fixed dimension vector, where each position reflects the probability of the input belonging to a certain class. In natural language processing tasks, $x$ typically encodes features such as words or characters occurring in a text that is being analyzed, e.g. for the purpose of assigning it to a given class. Therefore, some method to represent textual information in vector form is necessary.

A simple way of representing textual information is through the use of the Vector Space Model [14]. This approach represents each textual item as a sparse vector of identifiers, which can refer to individual words or word $n$-grams. For instance, when computing vector representations for words, one can use one-hot vectors of $V$ dimensions, where $V$ is the size of the considered vocabulary. Each word would thus be represented as a $V$-dimensional vector where all the vector’s values would be 0, except at the index of the word being represented, where the value would be 1.

Given the one-hot representation for words, a simple method of representing textual documents as fixed size vectors is to sum all the vector representations of the words composing the document:

$$d = \sum_{i=1}^{N} w_i$$

(2.1)
In Equation 2.1, $N$ is the number of words in the document being represented, and $w_i$ is the vector representation of word $i$. This operation will result in a sparse vector with dimensionality $V$, where the $i$th element will represent the term frequency of word $i$ in the document.

Instead of a simple sum, a weighted average can instead be considered, taking an Inverse Document Frequency (IDF) score for each word into account:

$$d = \sum_{i=1}^{N} \frac{\text{IDF}(i)}{N} \times w_i$$

(2.2)

The IDF score of a given word corresponds to a collection-level importance score, capturing the intuition that rare terms are more discriminative, and hence should have a higher importance:

$$\text{IDF}(i) = \log \left( \frac{|D|}{| \{ d' \in D | w_i \in d' \} |} \right)$$

(2.3)

In the above expressions, $|D|$ is the total number of documents in the collection $D$, and the quantity $| \{ d' \in D | w_i \in d' \} |$ corresponds to the number of documents that contain the term $w_i$.

Nowadays, one-hot representations for words are less popular. First of all, $V$ can be very large (as it is, most of the times), causing vectors to be too large to handle efficiently. Another major drawback is that the one-hot representation considers all words to be independent. The words *cat* and *feline* are as dissimilar as *dog* and *cake*. Similarity methods such as taking a simple cross-product or the cosine similarity between two one-hot vectors will yield 0, in the case of the aforementioned words.

One alternative is to consider a dense representation of textual information, in which words would be represented by lower-dimensional dense vectors. Over the years, many different approaches have been proposed for inferring dense vector representations of words with basis on large corpora. After training a model for building such representations, similar words (e.g., words like *dog* and *canine*) would have similar representations. These word representations are often called embeddings. Word embeddings are widely used today and are a part of most of the state-of-the-art models for natural language processing.

A popular method for unsupervised training of word embeddings is the word2vec [15] approach, which is based on the intuition that similar words will often appear in similar contexts. Inferring word2vec embeddings can be made with one of two different architectures: the skip-gram and the Continuous Bag of Words (CBOW) approaches, formalized by Mikolov et al. [16].

The skip-gram model aims at predicting a context given a center word, i.e., predict the surrounding words of some center word. Formally, given a sequence of words $w_1, w_2, w_3, \ldots, w_T$, the objective of the skip-gram model is to maximize the following average log probability:

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log(p(w_{t+j} | w_t)).$$

(2.4)

In the previous equation, $c$ is the context size, $T$ is the size of the textual corpus, in words, and $w_t$ refers
to a word at position $t$. The standard definition for $p(w_{t+j}|w_t)$ uses the softmax function, as shown below:

$$p(w_{t+j}|w_t) = \frac{\exp(V(w_{t+j})^T V(w_t))}{\sum_{w=1}^{W} \exp(V(w)^T V(w_t))}. \tag{2.5}$$

In the previous equation, $V(w_O)$ is a vector representation of an input word (i.e., the center word), $V(w_I)$ is a vector representation of an output word (i.e., a context word) and $W$ is the vocabulary size.

Alternatively, the CBOW approach is the opposite of the skip-gram approach, in the sense that the intention is to predict a center word, given a context of surrounding words. Given a sequence of words $w_1, w_2, w_3, \ldots, w_T$, the objective of the CBOW model is to maximize:

$$\frac{1}{T} \sum_{t=1}^{T} \log(p(w_t|w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c})). \tag{2.6}$$

In Equation 2.6, the variable $c$ is again the context size.

Given the embeddings for all the words in the vocabulary, simple approaches for representing entire documents involve taking a sum of the embeddings for the words within the document (i.e., as shown through Equations 2.1 or 2.2), or alternatively taking the average of the word embeddings.

### 2.2 Introduction to Neural Networks

Neural networks were proposed as a computational approach to making inferences from data, taking inspiration on the human nervous system. In general, neural networks can be seen as as nested composite functions that transform vector representations, and whose parameters can be trained directly to minimize a given loss function computed over the outputs and the expected results.

#### 2.2.1 The Perceptron

The simplest possible neural network, consisting of a single node, is commonly referred to as the Perceptron. The concept was introduced by Rosenblatt [17] for binary classification problems. Formally, the perceptron is a linear model where:

$$y = \text{signal}(x \cdot a + b) \tag{2.7}$$

In the previous equation, $\text{signal}(\cdot)$ is a function that returns -1 if its input is negative or 1 otherwise, while $x$ is the input vector and $a$ is a weight vector. Training the perceptron corresponds to iteratively picking an example from the training dataset and updating each weight from the weight vector $a$ according to the following equation:

$$a_i = a_i + \eta \cdot (\hat{y} - y) \cdot x_i \tag{2.8}$$

In this equation, $a_i$ is the $i$th element of $a$, $x_i$ is the $i$th element of $x$, $\hat{y}$ represents the true label for the data point being processed, $y$ is the predicted output, and $\eta$ is a small constant called the learning
rate. This training algorithm will converge to the correct classification of the training examples if $\eta$ is sufficiently small and if the training data is linearly separable (i.e., there exists a set of weights capable of correctly classifying all the instances in the training set).

### 2.2.2 The Multi-Layer Perceptron

Extending the aforementioned Perceptron model, the Multi-Layer Perceptron (MLP) is a simple neural network architecture which consists of a set of nodes forming the input layer, one or more hidden layers of computation nodes, and an output layer of nodes. The input signal propagates through the network layer-by-layer in a feed-forward manner, until it reaches the output node(s). Considering a single hidden layer, the corresponding computations can be written as shown in Equation 2.9.

$$y = \sigma(\sigma'(x \cdot A + a) \cdot B + b) \quad (2.9)$$

In the previous equation, $x$ is a vector of inputs and $y$ a vector of outputs. The matrix $A$ represents the weights of the first layer and $a$ is a bias vector for the first layer, while $B$ and $b$ are, respectively, the weight matrix and the bias vector of the second layer. The functions $\sigma'(\cdot)$ and $\sigma(\cdot)$ both denote an element-wise non-linearity, i.e. the activation functions respectively associated to nodes in the hidden layer, and in the output layer of the network (i.e., the equivalent to the $\text{sign}(\cdot)$ function in the case of the Perceptron model from Equation 2.7). The softmax function (i.e., a normalized exponential function that produces as output a probability distribution) is often used in the final layer of a MLP classifier, training the network to minimize a cross-entropy loss.

For more complex networks, such as MLPs, the training procedure involves an algorithm known as back-propagation, in combination with some variation of gradient descent optimization [18]. An optimization procedure that has been frequently used to train deep neural networks is the Adaptive Moment Estimation (Adam) algorithm [19]. Adam computes parameter updates leveraging an exponentially decaying average of past gradients, together with adaptive learning rates for each parameter. In practice, it performs larger updates for infrequent parameters, and smaller updates for frequent parameters.

While MLPs have been extensively used in NLP applications, the problems in this area often involve capturing regularities over structured data of arbitrary sizes (e.g., sequences of vectors representing individual words, i.e. word embeddings [20, 15]). In many cases, this means encoding the structure of a text as a fixed width vector, which can then be used for further processing. Other network architectures are thus commonly used in NLP, in order to transform a sequence of word embeddings $x_1, \ldots, x_T \in \mathbb{R}^d$ into a vector (e.g., a sentence representation) $s \in \mathbb{R}^h$. These architectures include Convolutional Neural Networks (CNNs) and different forms of Recurrent Neural Networks (RNNs).

### 2.2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) involve the application of $h$ filters (also named kernels), sliding them over the input sequence. Each filter performs a local convolution (i.e., an element-wise matrix
multiplication followed by a summation) on the sub-sequences of the input, to obtain a set of feature maps. Then, a global average- or max-pooling over time is performed to obtain a scalar, and the scalars from the \( h \) filters are finally concatenated into the sequence representation vector \( s \in \mathbb{R}^h \).

Assuming an input sequence \( x_1, \ldots, x_T \) (e.g., a sequence of embeddings for a document with \( T \) words), a convolution layer of width \( k \) works by moving a sliding window of size \( k \) over the sequence, creating several instances of windows \( w_i = [x_i; x_{i+1}; \ldots; x_{i+k-1}] \). A filter of size \( k \), i.e. a linear transformation followed by an activation function, is then applied to each window, resulting in \( m \) vectors \( p_1, \ldots, p_m \), where \( m = T - k + 1 \), or in case the sentence is padded with \( k - 1 \) words to each side, \( m = T + k + 1 \). Each vector \( p_i \) is defined as follows:

\[
p_i = \sigma(w_i \cdot A + a) \tag{2.10}
\]

In the previous equation, \( \sigma(\cdot) \) is an activation function that is applied element-wise, while \( A \) and \( a \) are parameters of the network. The \( m \) vectors can then passed through a max-pooling or an average-pooling layer and a final representation vector \( r \) is obtained. In case max-pooling is used, each element \( j \) of \( r \) is obtained as follows, where \( p_i[j] \) denotes the \( j \)-th component of \( p_i \):

\[
r[j] = \max_{1 \leq i \leq m} p_i[j]. \tag{2.11}
\]

### 2.2.4 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) can, in turn, be seen as time-dependent neural networks, which at time step \( t \) compute a hidden state vector \( h_t \), which is obtained by a non-linear transformation with two inputs – the previous hidden state \( h_{t-1} \) and the current word input \( x_t \):

\[
h_t = f(h_{t-1}, x_t) \tag{2.12}
\]

The most basic RNN is called the Elman RNN, corresponding to the following equation:

\[
h_t = \sigma(W_1 \cdot h_{t-1} + W_2 \cdot x_t) \tag{2.13}
\]

However, previous research has noted that the Elman RNN has difficulties in modeling long sequences. Extensions have been proposed to handle this problem, and two well-known examples are Long Short-Term Memory (LSTM) units [21] and Gated Recurrent Units (GRUs), originally proposed by [22]. Both these approaches involve different components, i.e. gating mechanisms, which interact in a particular way in order to combine previous states with the current inputs. For instance, GRUs correspond to the following equations:

\[
z_t = \varphi_R(W_z \cdot x_t + U_z \cdot h_{t-1} + b_z) \tag{2.14}
\]

\[
r_t = \varphi_R(W_r \cdot x_t + U_r \cdot h_{t-1} + b_r) \tag{2.15}
\]

\[
h_t = \varphi_h(W_h \cdot x_t + U_h \cdot (r_t \odot h_{t-1}) + b_h) \tag{2.16}
\]
\[ h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t \]  

(2.17)

In the previous equations, the operator \( \odot \) denotes the Hadamard product (i.e., the entry-wise product of two matrices), while \( x_t \) denotes the input vector at time step \( t \), and \( h_t \) denotes the hidden state at time step \( t \). The parameters \( W, U \) and \( b \) denote the different weight matrices and bias vectors, adjusted when training the model through back-propagation.

Notice that a GRU involves two gates, namely a reset gate \( r \), that determines how to combine the new input with the previous memory, and an update gate \( z \) that defines how much of the previous memory to keep around. If we set the reset gate to all ones, and the update gate to all zeros, we again arrive at the Elman RNN model that was discussed previously. The gating mechanism allows GRUs to better handle long-term dependencies. By learning the parameters for its gates, the network learns how its internal memory should behave, given that the gates define how much of the input and previous state vectors should be considered.

LSTMs are an alternative type of RNN, with more parameters than GRUs (e.g., they have an extra gate) although also reported to outperform them when more training data is available, and in tasks requiring modeling longer-distance relations [23]. LSTMs correspond to the following equations, where \( \odot, \varphi_g(\cdot), \varphi_h(\cdot) \) \( x_t, h_t, W, U \) and \( b \) have the same meaning as in the GRU equations.

\[ i_t = \varphi_g (W_i \cdot x_t + U_i \cdot h_{t-1} + b_i) \]  

(2.18)

\[ f_t = \varphi_g (W_f \cdot x_t + U_f \cdot h_{t-1} + b_f) \]  

(2.19)

\[ o_t = \varphi_g (W_o \cdot x_t + U_o \cdot h_{t-1} + b_o) \]  

(2.20)

\[ g_t = \varphi_h (W_g \cdot x_t + U_g \cdot h_{t-1} + b_g) \]  

(2.21)

\[ c_t = f_t \odot c_{t-1} + g_t \odot i_t \]  

(2.22)

\[ h_t = g_t \odot o_t \]  

(2.23)

Notice that LSTMs apply different gating mechanisms, namely through the use of (i) a forget gate that controls how much of the previous memory will be kept, (ii) an input gate that controls how much of the proposed update \( g_t \) should be kept, and (iii) an output gate that controls the output at time \( t \).

2.3 Deep Learning Methods for Natural Language Processing

Even though this dissertation focuses on the application of deep neural networks for the detection of fake news, deep learning is widely used for a variety of other challenging NLP tasks. This section offers an insight into the application of deep learning to two distinct NLP problems, namely text classification and natural language inference.
2.3.1 Text Classification

Text classification is one of the most popular problems in supervised machine learning today. The main purpose of text classification is to assign textual documents to a class label. A wide variety of deep learning approaches have been proposed, and some of these leverage interesting ideas regarding how textual information can be represented, which can also be useful for the problem of detecting fake news.

Both LSTMs and GRUs have been shown to achieve remarkable performance on many sequence classification problems, particularly when processing short texts. In addition, hierarchical arrangements of LSTMs or GRUs can be used for modeling long documents, accounting not only with word order but also with sentence structure [3, 4]. For instance, Yang et al. [3] proposed a hierarchical attention network for document classification, considering documents to be composed of sentences, and sentences as sequences of words. The authors used a bi-directional GRU (i.e., they concatenate the states produced by two GRUs to generate the result for each time step, with one GRU processing the input in the forward direction, and the other processing the input in reverse) to encode each word in a sentence, followed by an attention mechanism to weight relevant words in the aggregated representation of each sentence. The attention mechanism corresponds to the following equations:

\[
\begin{align*}
    u_{it} &= \tanh(W_h \cdot h_{it} + b_w) \\
    \alpha_{it} &= \frac{\exp(u_{it}^\top u_w)}{\sum_t \exp(u_{it}^\top u_w)} \\
    s_i &= \sum_t \alpha_{it} \times h_{it}
\end{align*}
\]

The vector \( h_{it} \) corresponds to the word representation produced by the bi-GRU at time \( t \), when processing sentence \( i \), while \( W_h \) and \( b_w \) are parameters to be learned. The vector \( u_{it} \) is a hidden representation of \( h_{it} \), \( u_w \) is a word-level context vector to be learned, \( \alpha_{it} \) is an importance weight, and \( s_i \) is the sentence vector, calculated as the weighted sum of the word annotations. To create a document vector, the same methodology can be applied. A bi-directional GRU encodes the sentence vectors, and an attention mechanism is used to determine the importance of each sentence, yielding a document vector. The equations for the document-level attention mechanism are analogous to those from Equations 2.24 to 2.26. To generate the final prediction, the document vector is processed through a softmax layer.

To evaluate the described model, Yang et al. experimented on six text classification datasets, consisting of Yelp reviews [24], IMDB reviews [25], Yahoo Answers [26] and Amazon reviews [26]. Their model performed significantly better than other models based on support vector machines leveraging n-gram features, LSTMs, or CNNs on the aforementioned datasets. Since the original work of Yang et al., other studies have successfully used the same model in other challenging text classification tasks, e.g. related to biomedical text mining [4].

Lin et al. [27] proposed a new approach to extract sentence embeddings from the individual word embeddings that compose a sentence, and applied the idea to several text classification tasks related to author profiling (i.e., given a tweet, determine the age range of the user who wrote it) and sentiment analysis (i.e., given a product review, determine the number of stars the user assigned to the product).
In brief, instead of using a vector, the authors represented the sentence embedding as a matrix, with each row of the matrix concerning different parts of the sentence. Firstly, they considered a sentence matrix \( S \) with \( n \) rows, constructed with every word embedding \( w_i \) at the \( i \)th row, i.e.,

\[
S = (w_1, w_2, \ldots, w_n).
\]

(2.27)

The word embeddings in \( S \) are processed through a bi-LSTM and a matrix \( H \) is constructed by stacking all the hidden states of the bi-LSTM, i.e.,

\[
H = (h_1, h_2, \ldots, h_n).
\]

(2.28)

While typical attention mechanisms only create one weight vector and consequently only focus on one aspect of the semantics of the sentence, this work used a matrix \( A \) with \( r \) rows of attention vectors, allowing the model to focus on \( r \) different semantic parts. More formally:

\[
A = \text{softmax}(W_{s2} \cdot \tanh(W_{s1} \cdot H^T))
\]

(2.29)

In the above equation, \( W_{s2} \) and \( W_{s1} \) are trainable parameters. The final representation of the original sentence is obtained by:

\[
M = AH
\]

(2.30)

In order to prevent the attention mechanism from creating similar weight vectors and therefore introducing redundancy, a penalization term was minimized during training, together with the model loss. The penalization term was defined as follows:

\[
P = \|(AA^T - I)\|_F^2.
\]

(2.31)

In the previous equation, \( \| \cdot \|_F \) represents the Frobenius norm and \( I \) is the identity matrix.

The proposed model was tested on an author profiling dataset\(^1\) consisting of 68,485 randomly selected tweets for training, 4,000 tweets for validation, and 4,000 tweets for testing. The age range was split into five classes: 18-24, 25-34, 35-49, 50-64, 65+. In terms of the sentiment analysis task, the authors used a Yelp dataset\(^2\) composed of 2.7 million review-star pairs. A total of 500,000 pairs were selected for training, 2,000 for validation, and 2,000 for testing. The proposed model was compared with two baselines: a bi-LSTM with max-pooling over the hidden states and feed-forward layers to output the classification, and a CNN also considering max-pooling and final feed-forward layers. The proposed approach was able to achieve an accuracy of 64.21% on the Yelp dataset and 80.45% on the author profiling dataset, scoring higher than both baselines on both datasets.

\(^1\)http://pan.webis.de/clef16/pan16-web/author-profiling.html

\(^2\)http://www.yelp.com/dataset
While RNNs are a more common approach for dealing with text processing, the use of CNNs is also frequent in the literature. Johnson and Zhang [28] proposed the Deep Pyramid CNN (DPCNN) model, i.e. a word-level deep CNN architecture for text categorization. The first component of the proposed architecture performed text region embeddings, which correspond to a generalization of the common word embeddings, in order to represent regions of one or more words. The region embeddings were enhanced with unsupervised embeddings obtained by two-views embedding training [29], which essentially corresponds to a function of a view $X_1$ that by definition preserves everything required to predict another view $X_2$, and that is trained on unlabeled data. This component was followed by a stack of convolution blocks (i.e., two convolution layers with 250 filters, each of size 3, and a shortcut connection between the two convolutions [30]), interleaved with max-pooling layers with size 3 and stride 2 for downsampling. Since the feature maps were kept constant, every downsampling operation reduced the data size by half, reducing the computation time by the same factor, and forming a pyramidal shape. The final pooling layer aggregated the internal data into one vector.

In order to test the DPCNN model, the authors considered text classification datasets from Yahoo, Yelp, and Amazon, compiled by Zhang et al. [26], and used several other baseline approaches such as the ShallowCNN by Johnson and Zhang [31] and the aforementioned Hierarchical Attention Network by Yang et al. [3]. The DPCNN was able to outperform all the other tested models, achieving the lowest error rate on the considered datasets.

### 2.3.2 Natural Language Inference

Another NLP task strongly related to the subject of this dissertation, and where deep learning methods have been extensively employed, is Natural Language Inference (NLI). In brief, NLI concerns with determining if a given hypothesis sentence $h$ can be inferred from a premise sentence $p$. Generalized versions of the task have also considered multiple possible relations between the hypothesis and the premise (e.g., relations like entailment, contradiction, or neutrality), and large datasets such as the Stanford Natural Language Inference (SNLI) [5] or the Multi-Genre Natural Language Inference (MultiNLI) [6] corpora have enabled significant progress in terms of deep learning methods for building effective semantic representations of natural language information (e.g., models trained for NLI tasks can provide rich domain-general semantic representations).

For instance, Conneau et al. [7] demonstrated that the supervised training of sentence embeddings, based on the aforementioned NLI datasets, can consistently outperform other state-of-the-art approaches for representing sentences in different NLP tasks (e.g., taking the average of word2vec or GloVe [20] embeddings, using unsupervised methods such as skip-thought sentence embeddings [32], or using other supervised models such as the paragraph-phrase approach described by Wieting et al. [33]). The model parameters obtained when solving the NLI problem can thus be used for initializing other NLP models, and this idea was also explored in the context of the work described in this dissertation. Conneau et al. have also advanced a generic architecture for addressing the NLI task, illustrated in Figure 2.1. In brief, the premise and the hypothesis can both be encoded by a CNN or an RNN,
creating a vector representation for each sentence. These representations are then matched in some way (e.g., through a concatenation of the vectors, or through an element-wise product), fed into a set of fully-connected layers, and finally processed through a softmax layer that generates a final prediction.

As a sentence encoder (i.e., to generate sentence representations $u$ and $v$, respectively from the premise and the hypothesis), Conneau et al. used a bi-directional LSTM, followed by a max-pooling operation over the sequence of output states. Then, the authors combined both these representations through the concatenation of $u$ and $v$, the element-wise product, and the difference between the vectors. Finally, the result was fed to multiple fully-connected layers, followed by a softmax layer.

Besides the aforementioned general model, several other NLI approaches have been proposed in the literature [34, 35, 36], and evaluated over the SNLI and MultiNLI corpora (e.g., the authors of SNLI maintain a leader-board on the website\(^3\) describing the corpus). For instance, Nie and Bansal [34] developed a model that scores an accuracy of 86.1% on the SNLI test set, an accuracy of 74.6% on the MultiNLI matched testing set, and 73.6% on the MultiNLI mismatched testing set. Following the same general approach from Figure 1, these authors used a sentence encoder that takes as input pre-trained GloVe [20] embeddings for the words in the sentence, and processes/refines the embeddings through three layers of bi-directional LSTMs with shortcut connections (i.e., the input sequences for the $i$-th bi-LSTM layer are the concatenated outputs of all the previous layers, plus the original word embedding sequence). The final representations of the premise and hypothesis are generated through a max-pooling operation between all the hidden states of the last bi-directional LSTM.

Chen et al. [37] proposed an NLI approach based on four different components: word embedding, sequence encoder, composition layer, and the top-layer classifier. The word embedding component obtains a representation for each word in the premise and the hypothesis by concatenating a character-level embedding with a holistic word-level embedding for each word in the sentence. The character-level embedding resulted from the application of a CNN with max-pooling to each character in a word. The holistic word-level embedding came from pre-trained GloVe vectors\(^4\). The sequence encoder component encodes both the premise and the hypothesis applying a stacked bi-LSTM with three levels and shortcut connections, obtaining a matrix of the hidden states for the premise and for the hypothesis. Since the premise and the hypothesis likely have different lengths, the hidden vectors obtained from the sequence encoder are composed in order to transform both sentences into fixed-length vectors. The composition layer is responsible for this task, computing an intra-sentence gated attention to obtain a

\(^3\)http://nlp.stanford.edu/projects/snli/
\(^4\)http://nlp.stanford.edu/projects/glove/
fixed-length vector for the premise and for the hypothesis. Exemplifying for the premise, the composition layer computes the following vector:

\[ v_p^g = \sum_{t=1}^{n} \frac{||i_t||_2}{\sum_{j=1}^{n} ||i_j||_2} h_p^t \]  

(2.32)

In the previous equation, \( n \) is the length of the premise, \( i_t \) is the input gate of the bi-LSTM in the top layer at time \( t \), \( ||*||_2 \) indicates the \( l^2 \)-norm, which converts vectors to scalars, and \( h_p^t \) is the hidden state of the stacked bi-LSTM at time \( t \). The authors also applied average-pooling and max-pooling to the hidden states to obtain the vectors \( v_p^a \) and \( v_p^m \), and the final fixed-length representation for the premise \( v^p \) was obtained concatenating \( v_p^g \), \( v_p^a \), and \( v_p^m \). The process for obtaining a fixed-length representation of the hypothesis \( v^h \) was analogous to the one explained above. Finally, the top-layer classifier concatenates \( v^p \), \( v^h \), the difference and the element-wise product between them, before sending the result through a feed-forward component with a final softmax layer. This approach achieved an accuracy of 85.5% on the SNLI test set, which was, at the time the article was released, the best reported result on this dataset. Regarding the MultiNLI corpus, the proposed model obtained an accuracy of 73.5% on the matched set, and an accuracy of 73.6% on the mismatched set.

Gong et al. [35] proposed the interactive inference network, a new architecture for the NLI task consisting of five modules: an embedding layer, an encoding layer, an interaction layer, a feature extraction layer, and an output layer. The embedding layer converts each word into a vector representation and constructs the sentence representation matrix. The encoding layer encodes the representations from the embedding layer by incorporating new information into the representations (e.g., through the use of a Bi-RNN to model the temporal interaction on both directions). The interaction layer combines the premise representation matrix and the hypothesis representation matrix (e.g., using a cosine similarity or a dot product) to create a word-by-word interaction vector. The feature extraction layer extracts semantic features from the interaction vector, e.g. using convolutions. The final component, the output layer, generates the class prediction (i.e., entailment, contradiction or neutral).

The authors also described a concrete implementation of the interactive inference network, which they called Densely Interactive Inference Network (DIIN). The embedding layer of the DIIN model used the concatenation of pre-trained GloVe embeddings, the result of a CNN with max-pooling over learned and randomly initialized character embeddings, and syntactical features such as one-hot part-of-speech tagging features. In the encoding layer, the premise representation and the hypothesis representation are passed through a two-layer highway network [38], generating new representations \( P^{hw} \) and \( H^{hw} \) for the premise and hypothesis, respectively. These representations are then passed through an intra-attention layer. Taking the premise as example, the intra-attention layer was implemented according to
the following equations:

\[ a_{ij} = \mathbf{w}_i^{\text{itrAtt}} \mathbf{P}_j^{\text{hw}} \mathbf{P}_i^{\text{hw}} \mathbf{P}_j^{\text{hw}} \circ \mathbf{P}_i^{\text{hw}} \mathbf{P}_j^{\text{hw}} \]

\[ \mathbf{P}_i^{\text{itrAtt}} = \sum_{j=1}^{p} \frac{\exp(a_{ij})}{\sum_{k=1}^{p} \exp(a_{k})} \mathbf{P}_j^{\text{hw}}, \quad \forall i, j \in [1, \ldots, p]. \]  

(2.33)

In the previous equations, \( p \) is the length of the premise, \( h \) is the length of the hypothesis, \( \mathbf{P}_i^{\text{itrAtt}} \) is a weighted summation of \( \mathbf{P}_j^{\text{hw}} \), \( \mathbf{w}_i^{\text{itrAtt}} \) are trainable weights, \( \circ \) is the element-wise multiplication, and \([;]\) is the vector concatenation. Following these operations, \( \mathbf{P}_i^{\text{hw}} \) and \( \mathbf{P}_i^{\text{itrAtt}} \) go through a fusion gating mechanism, implemented as follows:

\[ z_i = \tanh(W_1^T [\mathbf{P}_i^{\text{hw}}; \mathbf{P}_i^{\text{itrAtt}}] + b_1) \]

\[ r_i = \sigma(W_2^T [\mathbf{P}_i^{\text{hw}}; \mathbf{P}_i^{\text{itrAtt}}] + b_2) \]

\[ f_i = \sigma(W_3^T [\mathbf{P}_i^{\text{hw}}; \mathbf{P}_i^{\text{itrAtt}}] + b_3) \]

\[ \mathbf{P}_i^{\text{enc}} = r_i \circ \mathbf{P}_i^{\text{hw}} + f_i \circ z_i, \]  

(2.34)

In the previous equations, \( W_1^T, W_2^T, W_3^T, b_1, b_2, \) and \( b_3 \) are trainable weights, and \( \sigma \) is the sigmoid function. The same process is applied to the hypothesis to generate its representation \( \mathbf{H}_i^{\text{enc}} \).

In the interaction layer, an interaction vector is created as follows:

\[ i_{ij} = \mathbf{P}_i^{\text{enc}} \circ \mathbf{H}_j^{\text{enc}}, \forall i \in [1, \ldots, p], \forall j \in [1, \ldots, h]. \]  

(2.35)

In the above equation, \( \mathbf{P}_i^{\text{enc}} \) is the \( i \)th row vector of \( \mathbf{P}^{\text{enc}} \), \( \mathbf{H}_j^{\text{enc}} \) is the \( j \)th row vector of \( \mathbf{H}^{\text{enc}} \), and \( h \) is length of the hypothesis. For the feature extraction layer, the authors adopted the DenseNet model by [39] as a convolutional feature extractor. Finally, in the output layer, a linear unit was used for the final classification. The DIIN was able to achieve an accuracy of 88.9% on the SNLI dataset, 80.0% on the MultiNLI matched test set, and 78.7% on the MultiNLI mismatched test set, defining, at the time, new state-of-the-art results with these datasets.

At the time of writing this document, the state-of-the-art on the SNLI dataset belonged to Kim et al. [40], with their Densely-Connected Recurrent and Co-Attentive Neural Network, which scored an accuracy of 90.1% when ensembling the referred model. The architecture proposed by Kim et al. is composed of three main components: the word representation layer, the attentively connected RNN and the interaction and prediction layer.

In order to construct the representation for each word, the word representation layer constructs each word vector as the concatenation of the fixed GloVe embedding for that word, a trainable GloVe embedding, a character representation obtained when feeding character embeddings to a CNN with max-pooling, and a flag indicating whether the word being processed is present in the other sentence. The attentively connected RNN applies multiple bi-directional LSTMs on top of each other, together with an attention mechanism matching both sentences. This RNN can be described as follows:
In the previous equations, \( l \) represents the current RNN layer, while \( t \) represents the current time step. The RNN at layer \( l \) is \( H_l \). The attentive information of the \( i \)th word \( p_i \) from the sentence \( P \) against the other sentence \( Q \), of length \( J \), is \( a_{p_i} \). Given the increasing number of parameters as the network goes deeper, the authors employed an autoencoder in order to reduce the number of features, while maintaining the original information, which serves as a form of regularization.

For the interaction and prediction layers, Kim et al. first apply a max-pooling operation to the outputs of the final RNN layer, hence generating a vector representation for the premise and for the hypothesis. Then, a match vector is created through the usual operation of concatenating the premise representation, the hypothesis representation, their sum, difference and absolute difference. Finally, two fully-connected layers with a final softmax activation output the class prediction.

\[
\begin{align*}
  h^l_t &= H_l(x^l_t, h^l_{t-1}) \\
  x^l_t &= [h^l_{t-1}; a^l_{t-1}; x^l_{t-1}] \\
  a_{p_i} &= \sum_{j=1}^{J} \alpha_{ij} h_{q_j} \\
  \alpha_{ij} &= \frac{\exp(e_{ij})}{\sum_{k=1}^{J} \exp(e_{ik})} \\
  e_{ij} &= \cos(h_{p_i}, h_{q_j})
\end{align*}
\] (2.36)

### 2.4 Stance Detection in the Context of the Fake News Challenge

The Fake News Challenge\(^5\) (FNC-1) consisted of a competition for evaluating NLP methods designed to solve a stance detection problem between the headline of a news article, and the main body of text for a news article. Further details on this challenge are presented in Section 4.1. The leader-board of the competition points to several interesting methods, some of them reviewed in this section.

The organizers of the competition also provided a baseline consisting of a gradient boosting classifier leveraging hand-crafted features, such as multiple similarity scores between the headline and the body (e.g., based on \( n \)-gram overlap). Using 10-fold cross validation, the baseline achieved a weighted accuracy of 75.20%, following the evaluation metric detailed in Section 4.1 (i.e., an accuracy metric that gives extra weight to some of the classes).

The first place in the competition was obtained by the team **SOLAT in the SWEN\(^6\)**, which used an ensemble of two sub-models, each outputting predictions associated with a confidence score. The aggregated model made its decisions with basis on the weighted average of both sub-models, achieving a weighted accuracy of 82.02%. One of the sub-models was a gradient boosting classifier similar to that of the baseline, leveraging features like the number of overlapping words between the headline and the body, or similarity scores computed from word \( n \)-gram representations. The second sub-model

\(^5\)http://www.fakenewschallenge.org/

\(^6\)http://blog.talosintelligence.com/2017/06/talos-fake-news-challenge.html
was based on Convolutional Neural Networks (CNNs) for encoding the headline and the body, using pre-trained word2vec [15] embeddings. The outputs for the headline CNN and body CNN were concatenated and put through feed-forward layers.

In second place came Team Athene, with a weighted accuracy of 81.97%. Their model corresponded to an ensemble of five identical sub-models with randomly initialized parameters. These sub-models were Multi-Layer Perceptrons (MLPs) with seven hidden layers and a softmax at the end. Seven distinct feature types were considered, computed from either the headline, the body, or a combination of both. Examples of features include vectors of unigram occurrences, or the cosine distance between headline and body representations computed from a non-negative factorization of the unigram occurrence matrix. The final prediction came from hard voting between the five sub-models in the ensemble.

In third place, team UCL Machine Reading [41] achieved a weighted accuracy of 81.72%. This model was also fairly simple, consisting of a MLP with one hidden layer and a final softmax layer. The input feature vector resulted from a concatenation of Term-Frequency (TF) vectors built from the headline and the body, with the cosine similarity between TF-IDF vectors for the headline and the body.

When considering the top three teams, it is important to notice that all three leveraged hand crafted features, together with other neural approaches. In subsequent work after the publication of FNC-1 results, Tosik et al. [42] used only external features to feed both a gradient boosting classifier – directly for the four class classification problem – and an ensemble of gradient boosting classifiers, where the instances were first considered unrelated or related, and in the case of a related decision, they were disambiguated into one of the three remaining classes (i.e., agrees, disagrees, or discusses).

Specifically, Tosik et al. leveraged text similarity features such as $n$-gram or word overlap, or the cosine similarity between the TF-IDF vectors for the headline and the body, as well as other miscellaneous features such as the presence of several refuting words in the headline/body, or the length of the headline/body. After achieving a weighted accuracy score of 78.63% on the FNC-1 testing dataset, the authors conducted ablation tests and concluded that the most helpful features were the overlap features between $n$-grams and words of the headline and body, the refuting features based on a lexicon, and distance measures such as the cosine similarity between TF-IDF vectors of the headline and the body and the Word Mover’s Distance [43] between the headline and body. On the other hand, sentiment features and other distance scores, such as the Hamming distance, did not provide a positive contribution.

Pfohl and Legros [44] reported on other interesting submissions to FNC-1. These authors experimented with four different approaches, namely (i) a bag-of-words method, (ii) a basic LSTM approach, (iii) a method leveraging an LSTM together with an inner-attention mechanism, and (iv) a more sophisticated approach which the authors named conditional encoding LSTM with attention (CEA-LSTM).

The bag-of-words model was built by averaging embeddings for words occurring in the headline, for words occurring in the body, and then concatenating these vectors. This result is processed by a feed-forward network with a softmax output layer. Both the basic LSTM model, and the model combining an LSTM with an inner-attention mechanism, processed a concatenation of the headline and the article body to classify the stance. The inner-attention was computed over a window with the first 15 tokens of

http://medium.com/@andre134679/team-athene-on-the-fake-news-challenge-28a5cf5e017b
the concatenated text. Finally, the CEA-LSTM processed the headline and the body with two separate
LSTMs, using the final hidden state of the headline LSTM as the first hidden state of the body LSTM.
An attention mechanism operated over a window with the last 15 output states from the headline LSTM,
together with the final hidden state of the body LSTM. The best results reported by Pfohl and Legros
were achieved with the CEA-LSTM method, corresponding to a weighted accuracy of 80.8% (i.e., the
method based on LSTMs failed to outperform other simpler approaches to the task).

More recently, after the announcement of the winners for FNC-1, Bhatt et al. [8] described an
approach based on a MLP for combining neural representations, statistical summaries of the data, and
feature engineering heuristics. The neural representations leveraged skip-thought vectors [32] (i.e.,
sentence representations learned in an unsupervised manner, by encoding a sentence to predict the
sentences around it in a given text corpora) to encode the headline and the body. Given the encodings
for the headline and the body, the component-wise product and the absolute difference between the
vectors were computed and used as features. The statistical features correspond to unigram occur-
cences within the headline and the body, weighted according to TD-IDF. Finally, the external heuristic
features included the number of similar words in the headline and the body, the cosine similarity be-
tween vector encodings for the headline and the body, the number of matching \( n \)-grams, the sentiment
difference between the headline and the body, etc. Each of the three main sets of features is processed
through feed-forward layers before being combined and processed by a final softmax layer. This ap-
proach achieved a weighted accuracy of 83.08%, scoring higher than the winning team in FNC-1, and
corresponding to the current state-of-the-art in tests with the FNC-1 dataset.

Chaudhry et al. [45] described six different models for stance detection, which were also tested on
FNC-1 data. The first model was a baseline leveraging the Jaccard similarity between unigrams from
the headline and from the sentences in the body. The second was a MLP leveraging representations
for the headline and the body, e.g. obtained by averaging pre-trained GloVe embeddings. The third
model explored two independent LSTMs, one encoding the headline and another encoding the body.
The two final state vectors are passed to a softmax layer that generates the final prediction. The fourth
model also used an LSTM to encode the headline, but now its final state vector was used to initialize
another LSTM that encodes the body. The fifth approach augmented the fourth model by considering bi-
directional LSTMs. Finally, the sixth approach extended the fifth model with a self-attention mechanism
on top of the encoder LSTMs. All six models were tested on a custom training/testing data split (i.e.,
the official test set was not available at the time the paper was written, and thus the results cannot be
directly compared against those reported for the FNC-1 competition). The best results were achieved
by the fifth model (i.e., the bi-directional encoder), corresponding to a weighted accuracy of 95.3%.

Similarly to Tosik et al. [42], Bourgonje et al. [46] also addressed the separate tasks of (a) determining
whether a headline-body pair is unrelated or related and, in the latter case, (b) the task of determining
whether the body agrees, disagrees, or discusses the subject of the headline. Again, experiments
leveraged data from FNC-1. In the first task, the authors start by gathering two sets of \( n \)-grams, one for
the headline and another for the body. Then, the number of matching \( n \)-grams is multiplied by the length
and IDF value of the matching \( n \)-grams, and divided by the total number of \( n \)-grams. If the resulting
score is above a threshold, the pair is considered to be related. For the second task (i.e., the three class classification problem), Bourgonje et al. used a logistic regression classifier trained on features extracted from the headlines of the FNC-1 dataset. If the distance between the best and the second best scoring classes is below a given threshold, the authors use a separate binary classifier, trained on features from both the headlines and the bodies of FNC-1 news articles. Three binary classifiers were trained for this second-level model, namely one for discriminating between agrees and disagrees, one for agrees versus discusses, and another one for discusses versus disagrees. Bourgonje et al. evaluated the proposed approach by leveraging 50 different tests with random 90-10 splits of the training FNC-1 dataset, scoring a weighted accuracy of 89.59%. However, these results are again not directly comparable to those from the FNC-1 participants.

Zeng et al. [47] tested six different encoders in a neural method for the FNC-1 task, based on concatenating representations for the headline and the body, afterwards generating the final classification through a softmax layer. The first encoder consisted of separate bi-directional GRUs, one for the headline and another for the body, with representations obtained from the final GRU states. In the second approach, the headline is fed to a bi-directional GRU, whose final hidden state is used to initialize another bi-directional GRU that processes the body. The third encoder uses a bi-directional GRU to process the concatenation of the headline and the body text.

The fourth and fifth models both leverage attention mechanisms. Specifically, the fourth model compares every state of the bi-directional GRU processing the body with the final hidden state of the bi-directional GRU that processes the headline. The attention weights are generated as follows:

$$\alpha_i = \text{softmax}(q^\top \cdot W_s \cdot \tilde{p}_i).$$

(2.37)

In the previous equation, $q$ is the final state of the headline bi-directional GRU, $\tilde{p}_i$ is the $i$-th hidden state of the body bi-directional GRU, $W_s$ are trainable weights, and $\alpha_i$ is the attention weight of the $i$-th body token. The attention weights are used to compute a weighted sum of the hidden states from the body bi-directional GRU, and this result is then concatenated with the final hidden state of the bi-directional GRU encoding the headline.

The fifth model is very similar to the fourth, in this case computing the attention weights by comparing the hidden states of the body GRU with all the hidden states of the headline GRU, instead of only the final state. The final attention weight for each body hidden state is given by the maximum value over all the weights calculated for every hidden state of the headline GRU.

Finally, the sixth approach was named the Bilateral Multiple Perspective Matching model. First, the headline and the body are encoded using separate bi-directional GRUs. Then, a cosine similarity is calculated between every hidden state from the headline GRU, and every hidden state from the body GRU, as shown in Equation 2.38:

$$m_k = \text{cosine}(w_k \odot v_1, w_k \odot v_2).$$

(2.38)
In the previous equation, \( w_k \) are trainable weights, and \( v_1 \) and \( v_2 \) represent the hidden states to be compared. For every headline hidden state, several representations were created (i.e., as many \( m_k \) vectors as the length of the body). A max-pooling is then applied, in order to obtain a single vector for every headline state. For the body hidden states, the process is analogous. After this first layer of representations, a second layer again uses bi-directional GRUs to further process the headline and the body representations. The two resulting hidden states are finally concatenated, and fed to feed-forward layers that perform the classification.

The aforementioned six models were evaluated on a custom split of the FNC-1 dataset, and the fifth approach described by the authors (i.e., the attention model comparing the body hidden states with every headline hidden state) achieved the best results, with a weighted accuracy of 85.2%. Again, this result cannot be directly compared against those from the FNC-1 participants.

More recently, Mohtarami et al. [9] introduced a memory network for stance detection, evaluating it on the FNC-1 testing dataset. The proposed memory network can be seen as 6-tuple \( \{ M, I, F, G, O, R \} \), where the memory \( M \) is a sequence of representations, \( I \) is a mapping from inputs to their representations, \( F \) is an inference component that identifies the relevant parts of the input, \( G \) is a generalization component that updates the memory according to \( F \), \( O \) is an output generated for each new input given the current memory state, and finally \( R \) is conversion from \( O \) into a desired response format.

Given a news article and a headline, the input component \( I \) first converts the article into a 3D tensor \( d = (X, W, E) \), where \( X = \{x_1, \ldots, x_n\} \) are the paragraphs from the document, \( W = \{w_1, \ldots, w_v\} \) are the words that represent each paragraph \( x_j \), and \( E = \{e_1, \ldots, e_v\} \) are the word embeddings. Every \( x_j \) is separately processed through an LSTM and a CNN, generating representations \( m_j \) and \( c_j \). A similar procedure is considered for the headline, generating representations \( s_{lstm} \) and \( s_{cnn} \).

The inference component \( F \) takes the previously computed representations and generates two similarities, according to the following equations:

\[
\begin{align*}
\hat{p}_{lstm}^j &= s_{lstm}^T \cdot M \cdot m_j \quad (2.39) \\
\hat{p}_{cnn}^j &= s_{cnn}^T \cdot M' \cdot c_j \quad (2.40)
\end{align*}
\]

In the previous equations, \( M \) and \( M' \) are trainable similarity matrices. Additionally, another similarity vector \( \hat{p}_{tfsdf}^j \) is computed by applying the cosine similarity metric to TF-IDF representations for the news headline and for each sentence of the news article body.

The memory component \( M \) and the generalization component \( G \) update the \( m_j \) vectors according to the following equation:

\[
\tilde{m}_j = m_j \odot \hat{p}_{tfsdf}^j \quad (2.41)
\]

Then, the updated \( \tilde{m}_j \) and \( s_{lstm} \) vectors are used by the inference component \( F \) to compute separate \( \tilde{p}_{lstm}^j \) vectors, using the same procedure detailed above. The new \( \tilde{p}_{lstm}^j \) vectors are then used to update
the $c_j$ vectors, according to the following equation:

$$\tilde{c}_j = c_j \odot \tilde{p}^{lstm}_{j} \tag{2.42}$$

Finally, the updated $\tilde{c}_j$ representations in conjunction with $s_{cnn}$ are leveraged to compute new $\tilde{p}^{cnn}_{j}$ vectors, using the same procedure as explained above. All the aforementioned vectors are used in the output component $O$ to compute the following vector:

$$o = [\text{mean}({\{c_j\}}); \text{max}({\{p^{cnn}_{j}\}}); \text{mean}({\{p^{lstm}_{j}\}}); \text{max}({\{p^{lstm}_{j}\}}); \text{mean}({\{p^{tfidf}_{j}\}}); \text{max}({\{p^{tfidf}_{j}\}})] \tag{2.43}$$

In the response component $R$, the concatenation $[o; s_{lstm}; s_{cnn}]$ is fed to feed-forward layers, and a final softmax layer gives the prediction for the stance. In their tests, Mohtarami et al. achieved a weighted accuracy of 81.23%, again attesting to the difficulties associated to the application of modern neural approaches to the FNC-1 task.

### 2.5 Overview

This section reviewed fundamental concepts and state-of-the-art approaches for natural language inference and for the stance detection task of the Fake News Challenge (FNC-1). As the FNC-1 leader-board attests, the highest scores on the official testing dataset of the competition were obtained through the use of approaches heavily based on feature engineering, rather than deep learning architectures. Table 2.2 summarizes the results obtained with the FNC-1 methods surveyed in this chapter.

In the case of previous natural language inference approaches evaluated on the SNLI and MultiNLI datasets, neural approaches are more frequent. Table 2.1 provides a compact representation of the

<table>
<thead>
<tr>
<th>Method</th>
<th>SNLI</th>
<th>Matched</th>
<th>Mismatched</th>
</tr>
</thead>
<tbody>
<tr>
<td>Most Frequent Class [5]</td>
<td>34.3</td>
<td>36.5</td>
<td>35.6</td>
</tr>
<tr>
<td>Continuous BOW (Averaging Word Embeddings) [5]</td>
<td>75.3</td>
<td>65.2</td>
<td>64.6</td>
</tr>
<tr>
<td>BiLSTM [48]</td>
<td>83.3</td>
<td>67.5</td>
<td>67.1</td>
</tr>
<tr>
<td>Enhanced Sequential Inference Model [36]</td>
<td>88.0</td>
<td>72.4</td>
<td>71.9</td>
</tr>
<tr>
<td>Nie and Bansal [34]</td>
<td>86.1</td>
<td>74.6</td>
<td>73.6</td>
</tr>
<tr>
<td>Chen et al. [37]</td>
<td>85.5</td>
<td>74.9</td>
<td>74.9</td>
</tr>
<tr>
<td>Conneau et al. [7]</td>
<td>84.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Radford et al. [49]</td>
<td>89.9</td>
<td>82.1</td>
<td>81.4</td>
</tr>
<tr>
<td>Densely Interactive Inference Network [35]</td>
<td>88.9</td>
<td>80.0</td>
<td>78.7</td>
</tr>
<tr>
<td>Directional Self-Attention Encoders [50]</td>
<td>85.6</td>
<td>71.0</td>
<td>71.4</td>
</tr>
<tr>
<td>Compare-Propagate Alignment-Factorized Encoders [51]</td>
<td>85.9</td>
<td>78.7</td>
<td>77.9</td>
</tr>
<tr>
<td>Gumbel TreeLSTM Encoders [52]</td>
<td>86.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reinforced Self-Attention Network [53]</td>
<td>86.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distance-Based Self-Attention Network [54]</td>
<td>86.3</td>
<td>74.1</td>
<td>72.9</td>
</tr>
<tr>
<td>Densely-Connected Recurrent and Co-Attentive Neural Network [40]</td>
<td>90.1</td>
<td>80.6</td>
<td>79.5</td>
</tr>
</tbody>
</table>

Table 2.1: Accuracy of previous models for the natural language inference task, over the SNLI and MultiNLI testing datasets.
<table>
<thead>
<tr>
<th>Method</th>
<th>Weighted Accuracy</th>
<th>Per-Class Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>FNC-1 baseline[^6]</td>
<td>75.20</td>
<td>97.97</td>
</tr>
<tr>
<td>Baseline based on word2vec + hand-crafted features [8]</td>
<td>72.78</td>
<td>96.05</td>
</tr>
<tr>
<td>Baseline based on skip-thought embeddings [8]</td>
<td>76.18</td>
<td>91.18</td>
</tr>
<tr>
<td>Baseline based on TF-IDF vectors [8]</td>
<td>81.72</td>
<td>97.90</td>
</tr>
<tr>
<td>Best set of features from Tosik et al. [42]</td>
<td>78.63</td>
<td>97.98</td>
</tr>
<tr>
<td>Neural baseline based on bi-directional LSTMs [8]</td>
<td>63.11</td>
<td>78.27</td>
</tr>
<tr>
<td>Neural method from Mohtarami et al. [9]</td>
<td>78.97</td>
<td>–</td>
</tr>
<tr>
<td>Neural method from Mohtarami et al. + TF-IDF [9]</td>
<td>81.23</td>
<td>–</td>
</tr>
<tr>
<td>3\textsuperscript{rd} place at FNC-1 – Team UCL Machine Reading [41]</td>
<td>81.72</td>
<td>97.90</td>
</tr>
<tr>
<td>2\textsuperscript{nd} place at FNC-1 – Team Athene[^9]</td>
<td>81.97</td>
<td>99.25</td>
</tr>
<tr>
<td>1\textsuperscript{st} place at FNC-1 – Team SOLAT in the SWEN[^10]</td>
<td>82.02</td>
<td>98.70</td>
</tr>
<tr>
<td>State-of-the-art – Bhatt et al. [8]</td>
<td>83.08</td>
<td>98.04</td>
</tr>
</tbody>
</table>

Table 2.2: Results obtained with the testing split of the FNC-1 dataset.

state-of-the-art results relative to the SNLI and MultiNLI testing datasets.

The work presented in this dissertation leverages both feature engineering and deep learning approaches. Several ideas from the work surveyed in this chapter have been considered, namely the use of external features from participations in FNC-1, and also the exploration of the hierarchical nature of news articles – all the details are explained in Chapter 3.
Chapter 3

The Deep Neural Model for Fake News Detection

This chapter presents a deep neural network for detecting the stance of an article towards a headline. The proposed neural network architectures take inspiration on the models by Yang et al. [3], Nie and Bansal [34], and Conneau et al. [7], described on the previous chapter. Figure 3.1 provides an high-level overview of the proposed approach for the FNC-1 stance detection task.

The headline is processed through one of the sentence encoders described in Section 3.1, which outputs the corresponding representation. In turn, the body of the news article is processed through the document encoder described in Section 3.2, which leverages a hierarchical model combining the sentence encoder with an encoder for the sequence of sentences. A third branch in the model compares the headline with the first two sentences of the body, leveraging the sentence encoder to build the involved representations. The sentence encoder, which represents the basis of all three branches, is pre-trained with the SNLI and MultiNLI datasets, as explained in Section 3.3.

The representations for the headline, the entire body, and the first two sentences from the body, are matched through vector operations such as the element-wise product, the vector difference, or the concatenation. These results are then combined with external features computed from the headline and/or the body - see Section 3.4 - and the result is processed by two fully-connected layers, with the final one producing the stance classification.

The following sections detail the components of the architecture outlined above, with Section 3.5 presenting a summary of the chapter.

3.1 The Sentence Encoder

This section goes into detail on how to build a representation for each sentence given as input to the neural network. Two different encoders are presented: a relatively simple one, using a bi-directional RNN followed by a max-pooling operation and an attention layer, and a more complex approach leveraging two layers of bi-RNNs, shortcut connections, max-pooling, and attention.
The first proposed sentence encoder takes as input a sequence of \( w_l \) words (i.e., a sentence), with \( l \in [0, L] \) and where \( L \) is the length of the sequence, replacing each word by a pre-trained GloVe embedding [20]. The resulting matrix of embedded words \( E \) is processed through a bi-directional RNN, which generates a hidden state matrix \( H \) as follows:

\[
\begin{align*}
    h_{tf} &= \text{RNN}_{\text{forward}}(E_t), t \in [1, \ldots, L] \\
    h_{tb} &= \text{RNN}_{\text{backward}}(E_t), t \in [1, \ldots, L] \\
    h_t &= [h_{tf}; h_{tb}], t \in [1, \ldots, L] \\
    H &= [h_1, \ldots, h_L].
\end{align*}
\]

In the previous equations, the RNN function can be instantiated either with the LSTM or with the GRU equations, both shown in Chapter 2 and hence not repeated here. Finally, the RNN states \( h_t \) from matrix \( H \) are processed through a summarization mechanism, which outputs a single vector. In this work, the summarization mechanisms are either a max-pooling operation, and/or an inner-attention mechanism defined as shown in the next equations:

\[
\begin{align*}
    u_t &= \tanh(W \cdot h_t), t \in [1, \ldots, L] \\
    \alpha_t &= \frac{\exp(u_t^\top)}{\sum_t \exp(u_t^\top)}, t \in [1, \ldots, L] \\
    s &= \sum_t \alpha_t \times h_t, t \in [1, \ldots, L]
\end{align*}
\]

In the equations above, the matrix \( W \) corresponds to trainable weights, \( \alpha_t \) is an importance weight assigned to each hidden state, and \( s \) is the final representation of the input sentence.

In order to further augment the sentence encoder described above, two layers of bi-directional RNNs are also stacked, feeding the second bi-directional RNN layer with the concatenation of original embeddings \( E \) and the hidden states \( H \) from the first bi-RNN (i.e., considering shortcut connections between
3.2 The Hierarchical Approach for Encoding Documents

The neural encoder for the news article body leverages the sentence encoders from the previous section, combining a sentence encoder with the hierarchical approach described by Yang et al. [3]. Each sentence in the news article is thus encoded through the procedure described in Section 3.1, and the resulting sequence of sentence vectors is then processed through a similar encoder (i.e., either a stacked model with two bi-directional RNNs, considering shortcut connections, an inner-attention mechanism, and/or a max-pooling operation, or a simpler model considering one bi-RNN layer instead).

3.3 Model Pre-Training with the SNLI and MultiNLI Datasets

Instead of randomly initializing all the parameters of the proposed neural network architecture, the SNLI [5] and MultiNLI [6] datasets were used to pre-train the parts of the network that do (i) the encoding of sentences, and (ii) the matching of the headline against the first two sentences of the news article body. Previous studies have already attested to the benefits of similar pre-training procedures in other natural language processing tasks [7].

The headline is seen as the hypothesis sentence from the NLI task, and the first two sentences from the body are seen as the premise. The NLI model thus involves the same sentence encoder and a common matching strategy (i.e., combining the representations for the hypothesis and the premise through the element-wise multiplication, vector difference, and vector concatenation), combined with a final softmax layer that returns the NLI class. After pre-training this model, the final softmax layer is simply ignored and the remaining components are re-used on the complete FNC-1 architecture.
3.4 Combining the Representations with External Features

The previous section described the vector operations used for matching the headline against the first two sentences of the body. A similar approach is also used for matching the representation for the headline against the representation for the entire body of the news article. Thus, the vector resulting from the pre-trained NLI network, which matches the headline against the first two sentences, is concatenated with the vector encoding the entire body, and with the results from the element-wise product and the difference between the vectors representing the headline and the body.

Moreover, taking inspiration on the baseline model proposed by the organizers of FNC-1, the aforementioned representations built through the neural network are also combined with external features computed from the headline and/or the body. The following list enumerates the considered features:

1. The number of words in common between (a) the headline and the body of the news article, and (b) the headline and the first two sentences of the body;
2. Refutation features, based on the presence of refuting words, listed in a given dictionary, in the headline (e.g., words like deny, doubt, fraud or debunk);
3. Polarity features, based on the presence of words with high emotion/sentiment polarity in (a) the headline, (b) the entire body, and (c) the first two sentences;
4. The number of word tokens that are common to (a) the headline and the body of the news article, and (b) the headline and the first two sentences of the body;
5. The number of word $n$-grams that are common to (a) the headline and the body of the news article, and (b) the headline and the first two sentences of the body;
6. The soft cosine similarity [12], computed between representations leveraging word occurrences for (a) the headline and the entire body, or (b) the headline and the first two sentences of the body;
7. The BLEU score [10] computed between (a) the headline and the set of sentences from the body, and (b) the headline and the first two sentences of the body;
8. ROUGE scores [11] computed between (a) the headline and the set of sentences from the body, and (b) the headline and the first two sentences of the body;
9. The CIDEr similarity score [13], computed between (a) the headline and the set of sentences of the article, and (b) the headline and the first two sentences of the body;
10. The cosine similarity metric, computed between TF-IDF vector representations for the words occurring in the headline, and in the body of the news article.
11. A vector representation of the headline, with 50 dimensions, resulting from a Singular Value Decomposition (SVD) of a matrix with TF-IDF representations for the texts;
12. A vector representation for the body of the article, with 50 dimensions, resulting from a Singular Value Decomposition (SVD) of a matrix with TF-IDF representations for the texts;
13. The cosine similarity metric, computed between the SVD vectors for the headline and the body;

14. A vector representation for the headline, with 300 dimensions, produced by averaging the word2vec embeddings for the words occurring in the headline;

15. A vector representation for the body, with 300 dimensions, produced by averaging the word2vec embeddings for all the words occurring in the body of the article;

16. The cosine similarity metric, computed between averaged word2vec embeddings for the headline and body of the news article;

17. Sentiment polarity scores for the headline and the body of the news article, computed with basis on a given word polarity lexicon.

The first 5 features from the previous enumeration were taken from the official FNC-1 baseline system, provided by the organizers. Features 10 to 17 were taken from the system\(^1\) that won the FNC-1 challenge, developed by the team SOLAT in the SWEN. Finally, in addition to external features based on previous work within the context of FNC-1, features from previous work concerned with assessing text similarity (e.g., the soft cosine similarity metric), or concerned with the evaluation of NLP methods for automated translation (i.e., the BLEU score), text summarization (i.e., ROUGE scores), or caption generation (i.e., the CIDEr score) were also considered. The three aforementioned metrics all attempt to assess the similarity between a given input text and a set of reference texts. When computing them, the input text (i.e., the candidate) was considered to be the headline, and the set of reference texts was considered to be formed by each sentence that composes the corresponding news article. When considering only the first two sentences from an article, the scores between the headline (i.e., the candidate) and the concatenation of the referred first two sentences (i.e., a reference set with a single instance) were instead calculated.

BLEU [10] works by counting the number of matching unigrams between the candidate text and the references. For every unigram in the candidate, BLEU saves the maximum amount of times it appears in a reference. BLEU then takes the minimum values from the aforementioned counts, and the number of times that the corresponding unigram appears in the candidate, hence generating a value \(m\) for every unigram in the candidate. Finally, the BLEU score between the candidate and the references is computed by summing all the \(m\) values for every unigram, and then dividing the result by the number of unigrams in the candidate text.

ROUGE [11] is similarly based on matches between the candidate and the references. Different variations of the metric can be computed, and this work specifically considered ROUGE-1, ROUGE-2, and ROUGE-L. ROUGE-1 is based on the average of the number of overlapping unigrams between the candidate and the references. ROUGE-2 is similar, considering instead the number of overlapping bi-grams. ROUGE-L averages the number of longest common sub-sequences between the candidate and the references.

\(^{1}\)http://github.com/Cisco-Talos/fnc-1
CiDiEr [13] begins by representing each sentence (i.e., the candidate and the references) as a TF-IDF vector, with basis on the \( n \)-grams that compose it. The CiDiEr score between a candidate sentence and a set of reference sentences is then computed as follows:

\[
\text{CiDiEr}(c, S) = \sum_{n=1}^{N} w_n \times \left( \frac{1}{m} \times \sum_{j=1}^{\left| S \right|} \frac{g^n_c \cdot g^n_{sj}}{||g^n_c|| \times ||g^n_{sj}||} \right)
\]  

(3.8)

In the previous equation, \( c \) is the candidate sentence, \( S \) is the set of references, \( g^n_c \) is the TF-IDF vector representation for the \( n \)-grams occurring in the candidate, and \( g^n_{sj} \) is the TF-IDF vector for the \( j \)-th reference. The number of references is represented by \( m \), while \( w_n \) is a weight, and \( N \) is the maximum length of \( n \)-grams to consider. For this work, \( w_n = 1/N \) and \( N = 4 \).

Finally, the Soft Cosine Similarity (SCS) metric computes the similarity between two texts leveraging bag-of-words representations, being computed as follows:

\[
\text{SCS}(a, b) = \frac{a^\top \cdot M \cdot b}{\sqrt{a^\top \cdot M \cdot a} \times \sqrt{b^\top \cdot M \cdot b}}
\]  

(3.9)

In the previous equation, \( a \) and \( b \) are the representations for the input texts, and \( M \) is a relation matrix whose element \( m_{ij} \) expresses some relation between word \( i \) and word \( j \), hence guaranteeing that two texts without any word in common have a score above 0 as soon as they share related words. In this work, \( M \) is a sparse term similarity matrix computed from 50-dimensional GloVe word embeddings [20].

### 3.5 Summary

This chapter detailed the architecture of the proposed neural network for addressing the FNC-1 task. In brief, a sentence encoder based in RNNs processes all words in the input headline, and an attention mechanism weights the importance of each word. The result from the weighted average is combined with a max-pooling operation, to generate a vector representation for the headline. Similarly, a document encoder takes the article as input, and outputs a corresponding vector representation, making use of the sentence encoder to encode every sentence that composes the article and, analogously to the sentence encoder, using an attention layer and/or a max-pooling operation to compute the representation.

Section 3.1 presented the component of the neural network responsible for generating the representation for an input sentence, leveraging bi-directional GRUs or bi-directional LSTMs to handle the sequence of words that compose the sentence, together with max-pooling and an attention mechanism to give each word more or less importance.

Section 3.2 explained the hierarchical nature of the input data, and how a model architecture responsible for encoding a sequence of words into a sentence vector can be reused to encode a sequence of sentences into a document vector.

Considering the intuition that the first few sentences in a news article serve as a summary of the whole document, the sentence encoder is also used to create a matching vector between the headline
and the first two sentences in a document. Instead of randomly initializing the weights of the proposed model, the SNLI and MultiNLI datasets, created for the NLI problem, were instead considered within a pre-training procedure, as explained on Section 3.3.

Finally, given that previous work on the FNC-1 dataset emphasized the importance of feature engineering, Section 3.4 detailed a set of external features considered in the work presented in this dissertation, complementing the representations obtained through the neural network.
Chapter 4

Experimental Evaluation

This chapter describes the experimental evaluation of the proposed method. Section 4.1 overviews the different datasets that supported the tests, together with the considered experimental methodology. Section 4.2 presents the obtained results over the SNLI and MultiNLI datasets, while Section 4.3 details the obtained results over the FNC-1 task. Section 4.4 is focused on the attention mechanism proposed in this work, illustrating its application in helping to interpret the results of the model. Finally, Section 4.5 gives an overview of this chapter and the obtained results.

4.1 Dataset and Experimental Methodology

In total, three datasets were considered in this work. For the fake news detection task, which constitutes the core of this dissertation, the FNC-1 dataset released by the organizers of the Fake News Challenge was considered. For model pre-training, the SNLI and MultiNLI datasets were both used. The following subsections describe these corpora, and the custom evaluation metric proposed for the FNC-1.

Regarding the implementation of the neural architecture presented in this dissertation, the Keras\(^1\) deep learning framework was used, and the corresponding source code is available on GitHub\(^2\). The proposed neural network relied on representations for the word tokens based on pre-trained GloVe embeddings\(^3\) with 300 dimensions. Out-of-vocabulary words in data were represented by fetching the GloVe embedding of the most similar word, as given by the Jaro-Winkler similarity metric [55].

Given the implementations for RNNs available within Keras, every sentence (i.e., the headlines, the premise and hypothesis sentences from the NLI datasets, and the sentences within the body of news articles) was either zero-padded or truncated to have 50 tokens. Every news article body was also zero-padded or truncated to 30 sentences. All the tested RNN architectures had hidden states of 300 dimensions (i.e., the bi-directional RNN representations have 600 dimensions), and the feed-forward layers before the final softmax are composed of 600 and 300 neurons, respectively. The neural models were trained leveraging the Adam [19] optimizer with default parameters, for a maximum of 100 epochs.

\(^{1}\)http://keras.io/  
\(^{2}\)http://github.com/LuisPB7/fnc-msc  
\(^{3}\)http://nlp.stanford.edu/projects/glove/
4.1.1 The FNC-1 Dataset and Evaluation Metric

To support the training and testing of models participating in the FNC-1, the organizers of the challenge released training and testing datasets. The training dataset contains 49,972 instances (i.e., pairs of headline and body texts) classified with a stance. The labeled testing dataset was, in turn, released after the end of the competition, and it contains 25,419 instances. Table 4.1 presents elementary characterization statistics for the training/testing datasets provided by the FNC-1 organizers.

Due to the imbalance in the class distribution, and also due to the fact that the distinction between agree, disagree and discusses is much more relevant to fake news detection, the FNC-1 organizers suggested a weighted scoring system. If a test instance is unrelated and the model labels it correctly, the score will be incremented by 0.25. If the test instance is related, i.e., the correct label is either agree, disagree or discusses, then the score will be incremented by 0.25 if the model labels the pair with one of the previously mentioned labels. In case the model chooses the correct label of a related test instance, the score will be incremented by an additional 0.75. Summing up, the equation for the proposed weighted accuracy metric is as follows:

\[ Acc_{FNC} = 0.25 \times Acc_{Related,Unrelated} + 0.75 \times Acc_{Agree,Disagree,Discuss} \]  

(4.1)

4.1.2 The SNLI and MultiNLI Datasets

In order to provide a large-scale resource for NLI model development, Bowman et al. [5] created the Stanford Natural Language Inference (SNLI) corpus. The dataset contains 570,152 premise-hypothesis pairs labeled as entailment, neutral, or contradiction, and was two orders of magnitude larger than all other resources of the same type that were available at the time the dataset was released. The sentence pairs were obtained via crowdsourcing. The authors collected image captions from the Flickr30k corpus [56] and asked workers to provide a sentence entailing the caption, another contradicting the caption, and a final sentence being unrelated to the caption. In order to assess the quality of the collected pairs, the same crowdsourcing methodology was applied for about 10% of all instances, but this time the workers had to choose a label for a given premise-hypothesis pair. The authors also split the data onto training, testing, and validation sets, with 550,152 pairs going for model training, 10,000 pairs going for
More recently, Williams et al. [6] introduced the Multi-Genre Natural Language Inference (MultiNLI) dataset, a corpus derived from the SNLI dataset and designed for use in natural language understanding and natural language inference tasks, considering texts from multiple genres. The corpus is composed of 432,702 premise-hypothesis pairs, labelled with entailment (i.e., the hypothesis is true relatively to the premise), contradiction (i.e., the hypothesis contradicts the premise) or neutral (i.e., none of the previous two apply). The premises were derived from ten different sources of freely available text: face-to-face conversations, telephone conversations, 9/11 reports, travel guides, letters, five works by the Oxford University Press, articles from Slate magazine, articles from the Verbatim archives, government documents, and contemporary fiction. To collect hypothesis, the authors presented crowdworkers with a premise and asked them to compose a sentence entailing the premise, another contradicting the premise, and a final sentence being unrelated to the premise. This data collection method ensured that each label was distributed equally throughout the corpus. The MultiNLI dataset was split into training, testing, and validation data. From the ten different thematic subjects from which the data was collected, only five of them were present in the training data. As for validation and testing sets, all subjects were represented. Given this decision by the authors, two different testing/validation sets were considered: the matched datasets, that include subjects that also appear in the training dataset, and the mismatched datasets, whose subjects are not in the training dataset. Table 4.2 summarizes the main characteristics of the MultiNLI and SNLI datasets.

### 4.2 Experimental Results on the SNLI and MultiNLI Datasets

The complete model described in Chapter 3 involved pre-training the branch that matches the headline against the first two sentences of the body (and also the common sentence encoder used in the remaining branches) with data from the SNLI and MultiNLI corpora. Therefore, the performance of this sub-component was evaluated against previous alternatives for Natural Language Inference (NLI). The results are given in Table 4.3, presenting accuracy scores over the SNLI testing dataset, and over the
<table>
<thead>
<tr>
<th>Method</th>
<th>SNLI</th>
<th>MultiNLI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Most Frequent Class [5]</td>
<td>34.3</td>
<td>36.5</td>
</tr>
<tr>
<td>Continuous BOW (Averaging Word Embeddings) [5]</td>
<td>75.3</td>
<td>65.2</td>
</tr>
<tr>
<td>BiLSTM [48]</td>
<td>83.3</td>
<td>67.5</td>
</tr>
<tr>
<td>Enhanced Sequential Inference Model [36]</td>
<td>88.0</td>
<td>72.4</td>
</tr>
<tr>
<td>Nie and Bansal [34]</td>
<td>86.1</td>
<td>74.6</td>
</tr>
<tr>
<td>Chen et al. [37]</td>
<td>85.5</td>
<td>74.9</td>
</tr>
<tr>
<td>Conneau et al. [7]</td>
<td>84.5</td>
<td>–</td>
</tr>
<tr>
<td>Radford et al. [49]</td>
<td>89.9</td>
<td><strong>82.1</strong></td>
</tr>
<tr>
<td>Densely Interactive Inference Network [35]</td>
<td>88.9</td>
<td>80.0</td>
</tr>
<tr>
<td>Directional Self-Attention Encoders [50]</td>
<td>85.6</td>
<td>71.0</td>
</tr>
<tr>
<td>Compare-Propagate Alignment-Factorized Encoders [51]</td>
<td>85.9</td>
<td>78.7</td>
</tr>
<tr>
<td>Gumbel TreeLSTM Encoders [52]</td>
<td>86.0</td>
<td>–</td>
</tr>
<tr>
<td>Reinforced Self-Attention Network [53]</td>
<td>86.3</td>
<td>–</td>
</tr>
<tr>
<td>Distance-Based Self-Attention Network [54]</td>
<td>86.3</td>
<td>74.1</td>
</tr>
<tr>
<td>Densely-Connected Recurrent and Co-Attentive Neural Network [40]</td>
<td><strong>90.1</strong></td>
<td>80.6</td>
</tr>
</tbody>
</table>

Table 4.3: Accuracy on the natural language inference task, over the SNLI and MultiNLI testing datasets.

matched (i.e., including news subjects that also appear in the training dataset) and mismatched portions of the MultiNLI testing dataset.

Besides the complete sentence encoder described in Chapter 3, considering a stacked arrangement of bi-GRUs or bi-LSTMs with shortcut connections, other variants were evaluated, leveraging (a) a single bi-directional RNN layer, or (b) using only max-pooling or neural attention as the summarization method, instead of concatenating the results of max-pooling and attention. The best results on the SNLI dataset were obtained with a single bi-directional LSTM combined with max-pooling, whereas for the MultiNLI dataset the best results were achieved with a single bi-LSTM together with a combination of max-pooling and neural attention. Bi-directional GRUs achieved consistently worse results, and the combination of max-pooling with neural attention achieved almost similar results to the usage of max-pooling alone (i.e., better results when leveraging GRUs, and in some cases the combination with neural attention actually lead to worse results when leveraging LSTMs).

Although simpler NLI baselines are outperformed, the demonstrated results are still far from the current state-of-the-art. However, it should be noted that these results are not directly comparable to those of previous systems, given that the NLI model was trained on a combination of the training data from the SNLI and MultiNLI corpora. Moreover, results were not optimized for the NLI task, since the main concern was to pre-train the network for building effective representations of textual inputs. It is perhaps the case that a more sophisticated matching method would improve the results for the NLI.
Table 4.4: Results obtained over the testing split of the FNC-1 dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Weighted Accuracy</th>
<th>Per-Class Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>FNC-1 baseline</td>
<td>75.20</td>
<td>97.97</td>
</tr>
<tr>
<td>Baseline based on word2vec + hand-crafted features [8]</td>
<td>72.78</td>
<td>96.05</td>
</tr>
<tr>
<td>Baseline based on skip-thought embeddings [8]</td>
<td>76.18</td>
<td>91.18</td>
</tr>
<tr>
<td>Baseline based on TF-IDF vectors [8]</td>
<td>81.72</td>
<td>97.90</td>
</tr>
<tr>
<td>Best set of features from Tosik et al. [42]</td>
<td>78.63</td>
<td>97.98</td>
</tr>
<tr>
<td>Neural baseline based on bi-directional LSTMs [8]</td>
<td>63.11</td>
<td>78.27</td>
</tr>
<tr>
<td>Neural method from Mohtarami et al. [9]</td>
<td>78.97</td>
<td>–</td>
</tr>
<tr>
<td>Neural method from Mohtarami et al. + TF-IDF [9]</td>
<td>81.23</td>
<td>–</td>
</tr>
<tr>
<td>3rd place at FNC-1 – Team UCL Machine Reading [41]</td>
<td>81.72</td>
<td>97.90</td>
</tr>
<tr>
<td>1st place at FNC-1 – Team SOLAT in the SWEN [6]</td>
<td>82.02</td>
<td>98.70</td>
</tr>
<tr>
<td>Previous state-of-the-art – Bhatt et al. [8]</td>
<td><strong>83.08</strong></td>
<td>98.04</td>
</tr>
<tr>
<td>MLP with the considered external features</td>
<td>81.95</td>
<td>97.86</td>
</tr>
<tr>
<td>Bi-LSTM + max-pooling</td>
<td>81.29</td>
<td>96.99</td>
</tr>
<tr>
<td>Bi-GRU + max-pooling + attention</td>
<td>80.76</td>
<td>97.36</td>
</tr>
<tr>
<td>Bi-LSTM + max-pooling + attention (best encoder)</td>
<td>82.23</td>
<td>96.74</td>
</tr>
<tr>
<td>Stacked bi-LSTMs + shortcuts + max-pooling</td>
<td>82.16</td>
<td>96.13</td>
</tr>
<tr>
<td>Stacked bi-GRUs + shortcuts + max-pooling + attention</td>
<td>81.95</td>
<td>96.22</td>
</tr>
<tr>
<td>Stacked bi-LSTMs + shortcuts + max-pooling + attention</td>
<td>81.16</td>
<td>95.25</td>
</tr>
</tbody>
</table>

4.3 Experimental Results on the Fake News Challenge Dataset

Table 4.4 presents results obtained over the FNC-1 testing dataset, including results from (a) previous baseline methods leveraging feature engineering, (b) baseline methods using simpler neural networks, (c) the best submissions to the FNC-1 challenge, (d) the previous state-of-the-art on the FNC-1 testing dataset, (e) a Multi-Layer Perceptron (MLP) leveraging the complete set of external features outlined in Chapter 3, and (f) the proposed neural architecture, considering all the components and model pre-training with the NLI datasets. Each group in Table 4.4 corresponds to one of the previous six items, with the last group presenting the results obtained with different sentence encoders, i.e. considering only one bi-RNN or a stack of two bi-RNNs with shortcut connections, and a combination of max-pooling and neural attention. The best weighted accuracy was achieved with a sentence encoder leveraging a single bi-LSTM, similarly to what was obtained in the case of the NLI datasets. The complete neural model, leveraging the best sentence encoder, outperformed the baseline method based on a MLP with the considered set of external features (which also achieved competitive results, outperforming the third best team at the FNC-1 competition and performing similarly to the more complex neural network architecture from Mohtarami et al. [9], in terms of the weighted accuracy metric).

Several variations of the complete method introduced in Chapter 3 were also evaluated, using the best sentence encoder but without considering (a) model pre-training with the NLI datasets, (b) external features complementing the neural representations, (c) the matching between the headline and the first...
Table 4.5: Experimental results with different variations of the proposed method, over the testing split of the FNC-1 dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Weighted Accuracy</th>
<th>Per-Class Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best encoder with the complete architecture</td>
<td>82.23</td>
<td>96.74</td>
</tr>
<tr>
<td>Best encoder - pre-training</td>
<td>81.85</td>
<td>98.26</td>
</tr>
<tr>
<td>Best encoder - matching first sentences</td>
<td>83.38</td>
<td>97.27</td>
</tr>
<tr>
<td>Best encoder - external features</td>
<td>75.31</td>
<td>92.66</td>
</tr>
<tr>
<td>Best encoder - pre-training - first sentences</td>
<td>82.06</td>
<td>97.38</td>
</tr>
<tr>
<td>Best encoder - pre-training - first sentences - features</td>
<td>76.08</td>
<td>86.23</td>
</tr>
</tbody>
</table>

The proposed full model without the branch matching the headline against the first two sentences of the body slightly outperforms the previous state-of-the-art in the FNC-1 test split, obtaining a weighted accuracy score of 83.38%. The different ablation tests in which this branch has been removed seem to indicate that the matching against the first two sentences does not provide useful/additional information to the FNC-1 task. The results also show that both model pre-training and the external features are essential to achieving good results with the proposed neural network architecture, confirming previous studies that have shown that the FNC-1 task is challenging for methods based exclusively on representations built through deep neural networks.

When the weights of the network are not initialized with the weights from the NLI task (i.e., when there is no model pre-training), the weighted accuracy drops to 81.85% (or 82.02% if the matching between the headline and the first two sentences is also removed). If the external features are instead removed, the weighted accuracy drops to 75.31%, which is better than a baseline method leveraging a bi-directional LSTM [8] but only marginally better than the official baseline released by the organizers of FNC-1, and much worse than the winning entries at the challenge or the previous state-of-the-art result reported by Bhatt et al. [8].

### 4.4 Interpreting Results by Visualizing the Attention Weights

Besides evaluating model quality in terms of predictive accuracy, a different set of tests attempted to see if the neural attention weights associated to the sentence encoder, and to the hierarchical encoder for the news article body, could provide clues as to the rationale behind particular classification decisions. Figure 4.1 illustrates the attention weights computed by the neural network for two example instances from the FNC-1 testing dataset, with darker colors corresponding to the words/sentences to which the model assigned a higher weight. The two example instances shown in Figure 4.1 were correctly classified by the model (i.e., the best approach from Table 4.5), with one of them belonging to the agree class and the other to the disagree class. In both examples, the headline and the body of the news article contain words that can be highly indicative of the stance (e.g., words such as *fake*, *fraud* or *hoax*, present in the lexicons used to compute the refutation and polarity features).
The headline for the agree example refers to a fake tweet, with the word fake having a high weight. Some of the sentences in the body confirmed that the supposed author did not send the tweet, with words/expressions such as did not send or claimed also having a high weight. On what regards the disagree example, its headline simply states climate change is a hoax, while the last sentence of the corresponding news article (i.e., the sentence with the highest weight) denies the claim from the headline, containing the words climate change is not a hoax.

Figure 4.2 further illustrates the weights computed for the neural attention layers in association to specific words, specifically by presenting 2 violin plots with the distribution for the attention weights associated to the words fraud and debunk (i.e., words that belong to the FNC-1 refutation dictionary and that also appear frequently in the test instances). The plots contrast normalized values for the attention weights (i.e., the attention scores associated to words in the headline and in the body text were normalized using the min-max normalization scheme, together with the minimum and maximum attention weights that are seen in these two parts of the input) that are estimated by the classifier when analyzing test instances belonging to the disagrees versus the agrees category. As expected, the weights associated to aforementioned words tend to be higher in the case of documents belonging to the disagrees class, thus confirming that the attention layers are indeed providing interesting information that can support model interpretability and the design of user interfaces for manually inspecting the stance classification results.

In brief, the obtained results with the FNC-1 dataset confirm the importance of using external similarity features for tasks involving the modeling/matching of long pieces of text, and also that model pre-training can be a simple, yet effective, way of improving the representations build through deep neural networks for natural language processing.
Figure 4.2: Distribution for the neural attention weights associated to two distinct refutation words, within FCN-1 testing instances belonging to the disagrees versus the agrees class.

4.5 Summary

This chapter presented the experiments performed for the evaluation of the proposed approach for the FNC-1 task. Section 4.1 provided a detailed description of the datasets used for testing the model presented in this dissertation. Given the class imbalance in the dataset from FNC-1, this section also described a weighted accuracy metric proposed by the organizers and employed when testing.

Section 4.2 presented the experimental results of the models from Chapter 3 on the SNLI and MultiNLI datasets. While these results were relatively far from the current state-of-the-art, they are in line with some of the earlier approaches and this work argues that the considered models are adequate for learning good sentence encoders, to be used in the context of model pre-training. The main goal with the NLI datasets was to train models for inferring sentence representations from the NLI task, and transferring them onto the FNC-1 task. Section 4.3 presented validation results for this pre-training hypothesis, showing the obtained results on the FNC-1 data together with an ablation study in order to assess the contribution from the different components involved in the neural network architecture. A Bi-directional LSTM model, when pre-trained with the SNLI/MultiNLI datasets, achieved the highest weighted accuracy on the FNC-1, slightly surpassing the current state-of-the-art on the official FNC-1 dataset, reported by Bhatt et al. [8].

Finally, Section 4.4 presented some examples of how the attention mechanism can be used to provide some interpretability to the obtained results, by analyzing how different words and sentences were attended in each stance prediction.
Chapter 5

Conclusions and Future Work

This dissertation presented a deep learning approach to aid in the automatic detection of fake news. Concretely, this work tackled the Fake News Challenge (FNC-1), consisting of a stance detection task where the goal was to classify the kind of relation between the headline and the body of a news article. This chapter overviews the main contributions, and highlights possible directions for further work on this stance detection task.

5.1 Contributions

Experimental results confirmed what previous related work on the Fake News Challenge (FNC-1) task had already indicated – purely recurrent or convolutional neural approaches are not well suited for the FNC-1 stance detection task. Instead, this work balanced a neural component (i.e., recurrent neural networks) for building text representations, with careful feature engineering capturing text similarity (e.g., external features considered in previous FNC-1 studies), achieving results similar to those from previous studies. Furthermore, this dissertation also demonstrated how beneficial a strong approach for natural language inference can be, when attempting to solve stance detection leveraging model pre-training. This hypothesis was corroborated when initializing the weights of the stance detection architecture with the weights of the NLI counterpart. An attempt was also made in order to create an alignment between the headline and the first two sentences of the input article, but this did not demonstrate consistent improvements on the overall results.

Another part of the architecture which has proven to be of major importance was the neural attention mechanism. This allows the proposed model to pay more or less attention to individual words/sentences when generating representations for a sentence, or for a document. Neural attention contributes positively to the overall accuracy, also helping in terms of model interpretability.

In spite of the positive contributions present in this document, the practical usefulness of this method is still unclear. One possible application for assessing the veracity of an article would be to search for news articles that agree/disagree with the headline of the document being assessed. However, given the imbalance in the FNC-1 dataset that was used to train the proposed architecture, the model is not able to
perform a strong distinction between the agree and disagree classes, important for the aforementioned use case. Instead, FNC-1 models are trained mainly on unrelated instances (i.e., about 72% of the training dataset), which has few importance when assessing whether an article is true or fake.

5.2 Future Work

The results obtained when conducting the experiments presented in this dissertation evidenced problems dealing with the heavily imbalanced class distribution in the FNC-1 dataset. To further improve results, data augmentation procedures can be considered. One such procedure is the Synthetic Minority Over-sampling Technique (SMOTE), which works by synthesizing new instances for the minority classes, based on the $k$-nearest neighbors of each instance from a less represented class [57]. Such data augmentation techniques were not tested, since the formulation for the FNC-1 task explicitly stated that data augmentation was not allowed. Further options for dealing with imbalanced datasets include the definition of new loss functions. For example, Lin et al. [58] modified the standard cross entropy loss by introducing a scaling factor which allowed the model to focus on harder examples from less represented classes, down-weighting the prediction of the easier examples.

While the model presented in this dissertation leverages bi-RNNs, other types of recurrent nodes have been proposed that might provide an increase in classification accuracy. For instance, Bradbury et al. [59] proposed Quasi-Reccurrent Neural Networks (Quasi-RNNs). While regular RNNs have limited parallelism capabilities, since the outputs at every time step depend on the previous hidden state, CNNs are the opposite, being highly parallelizable but with no concept of memory. Quasi-RNNs are a hybrid of RNNs and CNNs, addressing the issues of both architectures, and allowing for parallel computation and a dependency on the overall order of the elements in the input sequence. Another idea for an RNN architecture is the Variational Bi-LSTM proposed in [60]. Bi-directional RNNs simultaneously model each input sequence in the forward and in the backward direction, allowing for a richer representation of the data being processed. However, the forward and the backward RNNs are usually trained separately. Based on the belief that the combination of the hidden states from both LSTMs can bring benefits, the authors used a variational auto-encoder [61] that takes as inputs the hidden states from the two LSTMs and creates a shared hidden representation of the VAE at each time step. These hidden states from the VAE are then used to reconstruct the hidden states of the LSTMs and are also fed forward to the next hidden state. This method creates a link between the two LSTMs that allows an improved learning of representations. Still other recent proposals for improved recurrent units include Nested-LSTMs [62], Cell-Aware Stacked LSTMs [63], or Extended LSTMs [64].

As this dissertation demonstrated, approaches for addressing the natural language inference task can be used to improve the obtained results on the Fake News Challenge through model pre-training. Therefore, exploring new sentence encoding architectures which perform better at NLI might be beneficial for the fake news detection task at hand. For this purpose, the official SNLI leaderboard\(^1\) contains a collection of models trained and tested on the SNLI dataset. It is then possible to take inspiration

\(^{1}\)http://nlp.stanford.edu/projects/snli/
from some of these sentence encoding-based models and apply these ideas on FNC-1, substituting the sentence encoder from Section 3.1. Neural architectures which leverage the hypothesis when modeling the premise (and vice-versa) are also relevant, although adapting one such model for the FNC task would require a different approach other than independently modelling the headline and the sentences that compose an article. One example of such an architecture is the Dependent Reading Bi-directional LSTM (DR Bi-LSTM) proposed by Ghaeini et al. [65]. The DR Bi-LSTM first encodes the premise taking the hypothesis into account, and vice-versa. This is achieved by processing the hypothesis with a bi-directional LSTM, and then processing the premise with a bi-LSTM initialized with the last hidden state of the hypothesis bi-LSTM. The hypothesis is encoded in an analogous manner. Furthermore, attention weights are calculated as the similarity of hidden states of the premise and the hypothesis, which will then be used to create an attentive representation of both the premise and the hypothesis. Both representations for each sentence (i.e., before and after the attention mechanism) are then matched and passed through an MLP, where the results are once again processed dependently with a bi-LSTM, with a max-pooling operation in the end. A final concatenation of both representations is passed through feed-forward, and the final softmax prediction is output.

While this work leveraged a lexicon based on refuting/polarity words such as fraud or hoax, the acknowledgement of another class of words, named discourse markers, might be relevant specially for the augmentation of the neural attention mechanism in the proposed approach. Discourse markers are a set of words used to express the discourse relation between either two sentences or the main clauses that compose a sentence. Common examples include but, and, or because. In their work, Pan et al. [66] attempted to solve the NLI task by first transferring knowledge from a Discourse Marker Prediction (DMP) task. In the DMP task, given a pair of sentences (or more accurately, a pair of clauses that compose the sentence), the goal is to predict the discourse marker the author used to connect those two texts. For this prediction task, the authors pass both clauses through a bi-directional LSTM with a max-pooling operation in the end, before combining both representations with the concatenation, sum, and element-wise product, and then feeding the resulting vector into feed-forward layers for final discourse marker prediction. For the NLI task, the premise and the hypothesis are first encoded by applying a bi-LSTM to the concatenation of their GloVe embeddings, character embeddings, part-of-speech and named-entity recognition tags, and matching features. Furthermore, the authors compute attended representations of the premise and the hypothesis, where the attention weights are created leveraging not only the aforementioned encoding of both sentences, but also their representation from the discourse marker prediction task. Pan et al. then generate additional intermediate representations of the premise and the hypothesis, matching the words that compose both sentences through operations such as the concatenation, difference and element-wise product, and feeding the result to another bi-LSTM. Finally, a feed-forward part with a final softmax layer outputs one of the three NLI classes.

Aside from different types of recurrent units or sentence encoding architectures, another option to consider for improving the proposed neural network is to enhance the embedding layer. For instance, when handling out-of-vocabulary words (i.e., words for which there is not a pre-trained GloVe representation), or even for enriching existent word embeddings, different mechanisms can be considered which
generate new word representations from individual characters or character n-grams [67, 68, 69]. This might allow rare words or misspelled words to have a meaningful representation.

In addition to the Fake News Challenge, several other fact checking tasks are currently being organized, which may serve as a logical direction for future work. Several researchers from the University of Sheffield, Imperial College London, and Amazon Research Cambridge are organizing a workshop on Fact Extraction and VERification\(^2\) (FEVER). Given a textual claim, the task consists of extracting evidence (i.e., sets of sentences from Wikipedia) that support or refute the claim. Given these sets of evidence, the system must then make a final decision, and classify the input claim as *Supported*, *Refuted*, or *NotEnoughInfo*. More information regarding the task, the provided datasets for model training and testing, and some baseline approaches can be found in a paper from Thorne et al. [70].

Besides leveraging textual information and the stance of headlines towards longer pieces of text, fake news detection can also benefit from the combination of text mining methods, such as the one advanced in this dissertation, with other types of approaches. For example, some previous studies have noted that fake news stories are often relatively simple (i.e., fake stories are usually *flatter* than ordinary news stories), using exaggeration in a way that makes it easy to detect and follow them [71], or mentioning facts that can easily be disproven through inconsistencies against external sources of information.\(^3\). Fake news stories also have particular geographical diffusion patterns [71], with rapid reproduction and elevated mutation rates (i.e., fake news rise and fall in weeks and often in days). Future work in the area can perhaps consider the combination/extension of text mining methods with other types of external features, capturing different characteristics of news articles.

---

\(^2\)http://fever.ai/  
\(^3\)http://fakenews.publicdatalab.org/
Bibliography


