



CHALMERS
UNIVERSITY OF TECHNOLOGY



UNIVERSITY OF GOTHENBURG

Combating Fake News with Stance Detection using Recurrent Neural Networks

Master's thesis in Algorithms, Languages and Logic

Alexander Ågren
Christian Ågren

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2018

MASTER'S THESIS 2018

Combating Fake News with Stance Detection using Recurrent Neural Networks

ALEXANDER ÅGREN
CHRISTIAN ÅGREN



Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2018

Combating Fake News with Stance Detection using Recurrent Neural Networks
ALEXANDER ÅGREN
CHRISTIAN ÅGREN

© Alexander Ågren & Christian Ågren, 2018.

Supervisors: Alexander Schliep & Prasanth Kolachina, Department of Computer
Science and Engineering
Examiner: Richard Johansson, Department of Computer Science and Engineering

Master's Thesis 2018
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in L^AT_EX
Gothenburg, Sweden 2018

Abstract

Comparing the attitude that different news organizations have towards a certain claim or topic is an important part of the procedure used by human fact-checkers today for assessing the veracity of a news story reporting about the issue. In this thesis we focus on automating the challenging task of stance detection in the news domain, specifically determining the relative stance of an article towards a claim stated in an associated headline, making use of the labelled dataset delivered for supervision in the first stage of the Fake News Challenge. While the most successful approaches in this domain have used complex ensemble classifiers employing large sets of hand-engineered features, their performance is just marginally better than a simple bag-of-words model deriving only lexical similarity. Our approach makes use of recurrent neural networks that read headlines and articles word-by-word. The models we implement are comparable to the state-of-the-art systems, however, observing that severe overfitting occurs.

Keywords: fake news, stance detection, natural language processing, machine learning, recurrent neural networks.

Acknowledgements

TBA

Alexander Ågren & Christian Ågren, Gothenburg, January 2018

Contents

List of Figures	xi
List of Tables	xiii
1 Introduction	1
1.1 Background	1
1.2 Aims	2
1.3 Problem Formulation	2
1.3.1 Representation	3
1.3.2 Class Imbalance	4
1.4 Contribution	4
1.5 Delimitation	4
1.6 Outline	4
2 Theory	5
2.1 Representation	5
2.1.1 Text Documents	5
2.1.2 Words	7
2.2 Recurrent Neural Networks	8
2.2.1 Long Short-Term Memory	8
2.2.2 Gated Recurrent Units	9
2.2.3 Bidirectional Units	9
2.3 Multi-Layer Perceptron	10
2.4 Training	11
2.4.1 Loss Function	11
2.4.2 Gradient Descent	11
2.4.3 Generalization	12
2.5 Class Imbalance	13
2.5.1 Performance Metrics	13
2.5.2 Balancing the Class Distribution	14
2.5.3 Cost-Sensitive Learning	14
2.5.4 Ensemble of Classifiers	15
3 Fake News Challenge Stage 1	17
3.1 Dataset	17
3.2 Evaluation Metric	18
3.3 Baseline	18

3.4	State-of-the-Art	19
3.5	Related Work	20
3.5.1	Features for Stance Classification	20
3.5.2	Recognizing Textual Entailment	21
4	Models	23
4.1	Parallel Encoder	23
4.2	Conditional Encoder	23
4.3	Neural Attention	25
5	Experiments and Results	27
5.1	Representation and Classifier	27
5.1.1	Representing Articles	29
5.1.2	Employing Attention	29
5.2	Learning With Limited Supervision	30
5.2.1	Dropout Words	31
5.2.2	Dropout Sentences	31
5.2.3	Synonym Replacement	32
5.2.4	Transfer Learning	32
5.2.5	Regularization	33
5.3	Addressing the Class Imbalance Problem	33
5.3.1	Balancing the Class Distribution	34
5.3.2	Cost-Sensitive Learning	34
5.3.3	Ensemble of Classifiers	34
5.4	Summary	35
6	Discussion and Analysis	39
6.1	Representation and Classifier	40
6.2	Learning With Limited Supervision	41
6.3	Approaching the Class Imbalance Problem	41
6.4	Qualitative Analysis	42
6.4.1	Attention Models	43
6.4.2	Reducing and Increasing Bias	45
6.4.3	Difficulties and Limitations	45
6.5	Model Selection	46
7	Conclusion	53
	Bibliography	55
A	Appendix 1	I
A.1	Refuting Words	I
A.2	Hedging Words	I
A.3	Stop Words	I

List of Figures

4.1	Illustration of the parallel encoder. Nodes with superscript s and t represent a sentence s and another sentence t respectively. The final states of the two RNNs, \mathbf{h}_M^s and \mathbf{h}_N^t , correspond to the semantic representations of the two sentences. The concatenation of the final states is denoted $[\mathbf{h}_M^s, \mathbf{h}_N^t]$	24
4.2	Illustration of the conditional encoder. Nodes with superscript s and t represent a sentence s and another sentence t respectively. The final state of the second RNN, \mathbf{h}_N^t , represents the entailment relationship between the two sentences.	25
6.1	The conditional encoder based classifier, encoding the headline conditioned on the representation of the article, predicts the headline-article pair shown in this figure as <i>disagree</i> while the actual label is <i>unrelated</i> . If the word "authentic" is removed from the article, the classifier predicts the pair as <i>unrelated</i> . Removing "gender" in the headline keeping all words in the article, makes the classifier believe the pair is <i>unrelated</i> . What we see here is that the model generalizes poorly when removing a certain word in a headline or article. This is an indication that the model suffers from overfitting.	43
6.2	Illustrating weighted sentences and words in a headline-article pair from the FNC-1 test set, employing an attention model on top of the conditional encoder. The pair is labelled <i>agree</i> and the predicted label agrees with the actual label.	44
6.3	Plotting the training procedure of an RNN based model for 30 epochs, where (a) and (b) shows the relative fnc-score and the accuracy on the test set, the official hold-out set and the training set for each epoch and (c) shows the cross entropy loss on each training epoch.	48
6.4	Plotting the training procedure of an RNN based model for 30 epochs, where (a) and (b) shows the relative fnc-score and the accuracy on the test set, the non-overlapping hold-out and training sets for each epoch and (c) shows the cross entropy loss on each training epoch.	51

List of Tables

2.1	Confusion matrix for a binary-class classification problem.	13
3.1	Headline and article pairs of different labels. The example is taken verbatim from the FNC-1 official site [35].	18
3.2	Top 10 results in the FNC-1 competition	19
5.1	% FNC-score on the hold-out set and the test set for different model setups and activation functions.	28
5.2	% FNC-score on the hold-out set and the test set investigating different methods to improve the learning algorithm.	31
5.3	% FNC-score on the hold-out set and the test set addressing the class imbalance problem.	33
5.4	Confusion matrix on the hold-out set for classifier (i) in the ensemble classifier.	35
5.5	Confusion matrix on the hold-out set for classifier (ii) in the ensemble classifier.	36
5.6	Confusion matrix on the hold-out set for classifier (iii) in the ensemble classifier.	36
5.7	Confusion matrix on the test set for classifier (i) in the ensemble classifier.	36
5.8	Confusion matrix on the test set for classifier (ii) in the ensemble classifier.	36
5.9	Confusion matrix on the test set for classifier (iii) in the ensemble classifier.	37
5.10	Confusion matrix on the test set for the ensemble classifier.	37
5.11	Confusion matrix on the hold-out set for the conditional encoder based classifier, $A \rightarrow H$, representing articles as sets of sentences, trained with dropout on sentences in an article, setting the probability of keeping a sentence to 0.6, and considering the cost equal for all types of mis-classifications.	37
5.12	Confusion matrix on the test set for the conditional encoder based classifier, $A \rightarrow H$, representing articles as sets of sentences, trained with dropout on sentences in an article, setting the probability of keeping a sentence to 0.6, and considering the cost equal for all types of mis-classifications.	38
5.13	Confusion matrix on the test set for the FNC-1 baseline.	38

5.14 Confusion matrix on the test set for the state-of-the-art bag-of-words model.	38
--	----

1

Introduction

In this thesis we investigate whether machine learning methods can help to detect fake news. The New York Times narrowly defines fake news as "a made-up story with an intention to deceive" [41]. As a further matter, fake news is often the result of an interest to mislead people for some secondary gain [35]. For example, spreading false information about opponents can strengthen a certain political interest. Experts claim that made-up stories about Hillary Clinton were one of the reasons why she lost against Donald Trump in the last US presidential election [3]. Misinformation and influence campaigns have been reported in other countries as well [8]. Reportedly Russia attempted to influence the UK referendum about leaving the European Union by posting more than 45,000 messages on Twitter [1]. The trend of wide-spread use of mobile devices lead to an increasing proportion of the population receiving their news online, often in near real-time and often from novel news sources. As a consequence, there is a high risk of falling victim to fake news today. Many people are not aware that the news they read is fabricated as it is often a tedious and complex task to determine the veracity of a news story. Indeed, this can be a challenging task even for trained fact-checkers [5] [35]. In a research poll made in the US, about 64% of the adults answered that made-up news stories cause "a great deal of confusion about the basic facts of current events" [7] [35]. According to experts, the larger problem caused by fake news is that people start doubting real news [41].

1.1 Background

Addressing the fake news problem, the fact-checking organization Full Fact aims to implement machine learning and artificial intelligence to improve detection of fake news using automatic tools [5]. For this purpose, a competition called the Fake News Challenge stage 1 (FNC-1) was recently organized by Dean Pomerleau, an adjunct faculty member at Carnegie Mellon University, and Delip Rao, founder of Joostware AI Research Corporation [35]. Along with the competition, a large dataset containing labelled headline and article pairs was released. The dataset allows for studying two tasks, to decide if headlines and articles are related or not and to determine the attitude expressed in an article towards a certain claim given in an associated headline. In natural language processing, the latter task is referred to as stance detection. More specifically, the problem of stance detection is to infer from a source text whether the author is *for*, *against* or *neutral* to a subject given in a target text.

The organizers of the FNC-1 competition argue that the best way to address the fake news problem at this stage is to look at what parts of the procedure of assessing the veracity of a news article can be broken down into steps and automated assisting human fact-checkers [35]. An important first step in the fact-checking pipeline is to gather relevant background information and understand what different news organizations report about an issue including all sides there are to it. More specifically, the organizers explain that an automated tool for stance detection can help detect fake news as follows. First, given a claim one can find all top articles that agree, disagree or discuss the claim. In addition, such a tool can help identify specific arguments for and against the claim in these articles. Second, if many news articles from highly credible sources argue against a given claim, then the claim in question is likely not true. Conversely, if a claim is supported in a single news article from a low-credibility source and there is no mention of the claim in any article from a high-credibility source, one may draw the conclusion that the claim in question is probably not true.

Having an automated tool for detecting if a headline is unrelated to an associated article or not, can help identify fake news seeing that fabricated stories are often given headlines aiming to attract readers rather than reflecting the content of the articles.

1.2 Aims

- A) In the first part of this thesis, we investigate the efficiency of representing news articles using recurrent neural networks for the problem of detecting the relationship between a headline and its corresponding body text. Studying this, we train and evaluate a classifier on the dataset provided for the FNC-1 competition [35]. More specifically, we aim for the classifier to accomplish the following tasks.
- Determine whether a news article headline and the corresponding body text is related to each other or not. That is, classify a headline and article pair into one of the two categories *unrelated* or *related*.
 - Given a headline and an article that is related to each other, classify the stance of the article relative to the claim stated in the headline into one of the three categories *agree*, *disagree* or *discuss*.
- B) In the second part of this thesis, we attempt to solve the class imbalance problem present in the dataset, aiming for a classifier built for the tasks listed in A) to perform well in other domains.

1.3 Problem Formulation

Achieving our goals required addressing two subproblems: (i) A model must learn to distinguish and capture relevant information in long and content-sparse news

articles for classifying the relationship between the headline and the corresponding body text, and (ii) the classifier needs to account for shifts in class distribution. In this section, these subproblems are described in more detail.

1.3.1 Representation

To train a classifier for determining the relationship between a headline and an associated body text, a meaningful representation of news articles is needed. Typically, the headline of a news article is a shorter text concisely describing a topic or claim. On the other hand, the body text is virtually always a longer text document elaborating on the topic or arguing for the claim in the corresponding headline. A text document in this context consists of a sequence of paragraphs of several sentences each. There are various ways of representing text documents for classification tasks. The current state-of-the-art models for the FNC-1 dataset represent the headline and the body text using statistical methods, namely n -gram models and topic models. These models efficiently transform a news article into a fixed-size vector representation, considering which sequences of n words appear and how frequently they appear in both the headline and the body text.

In this thesis we model the headline as a sequence of words using a recurrent neural network and we model the associated body text either as a sequence of words or as a set of sentences, where a sentence is a sequence of words, using another recurrent neural network. A recurrent neural network considers all words and the order in which they appear in a text, taking long-term dependencies into account. An example of a long-term dependency is the use of "he" or "she" referring to a subject mentioned earlier. We believe that both the order in which words appear and long-term dependencies are important aspects to consider for the task of stance detection. For instance, a recurrent neural network can capture how the word "not" alters, depending on where it appears in the text, the attitude towards some event involving a certain subject referenced via long-term dependencies. Such textual relations might be harder to capture using a limited n -gram model.

Bowman et al. demonstrated that recurrent neural networks can learn efficient semantic representations of sentences for the similar task of recognizing textual entailment [10] [37]. Indeed models based on recurrent neural networks are currently the state-of-the-art for this task [44]. However, the dataset which is used for supervision consists of sentence pairs of high quality while the FNC-1 dataset used in this thesis consists of real news articles. Representing large news articles is a key challenge in this thesis since in general it is problematic for a recurrent neural network to capture all the important information in larger input sequences. Also, a news article is typically content-sparse [46]. This means that much of the information in a news article is considered noise and does not contribute with any useful content for making a certain classification. Therefore a model for a headline and body text pair needs to distinguish only the relevant information.

1.3.2 Class Imbalance

In the FNC-1 dataset, the set of headline and body text pairs that are related to each other is a minority compared to the set of pairs labelled *unrelated*. Training a classifier on a dataset with an imbalance in class distribution, the classifier tends to be biased towards the majority classes and therefore shows poor performance on instances belonging to the minority classes. The problem is referred to as the class imbalance problem [21]. Furthermore, we cannot assume the class distribution of news articles in another domain such as in a real world application is the same as in the development domain. Consequently, a classifier needs to generalize well on shifts in the class distribution. This issue becomes even more difficult in an imbalanced development domain as a classifier needs to perform well on minority classes, considering that in a shifted class distribution a minority class can become a majority class.

1.4 Contribution

Our contribution with this thesis is an evaluation of the efficiency of representing news articles using recurrent neural networks for the task of determining the relationship between a headline and the corresponding body text. For the subtask of detecting if a headline and body text pair is related or not, we find that the recurrent neural network models are less efficient on unseen examples than simple static representations constructed using statistical methods. However, for the stance detection subtask we find these comparable to the state-of-the-art.

1.5 Delimitation

The rules for the FNC-1 competition prohibits the use of any labelled dataset for supervision other than the one provided. Thus we decided to limit our study to only explore supervision using the FNC-1 dataset.

1.6 Outline

The remainder of this thesis is structured as follows. First in Chapter 2 we provide the theory behind the machine learning methods we make use of in this work. Thereafter in Chapter 3 we examine the FNC-1 dataset and describe the FNC-1 evaluation metrics. We also present the current state-of-the-art and study previous research relevant for this thesis. The recurrent neural network based models we investigate are presented in Chapter 4. Thereafter in Chapter 5 we describe the experiments conducted and report the results obtained. In Chapter 6 we discuss and analyze the methods investigated and lastly, in Chapter 7, we summarize our accomplishments and suggest how to extend this study.

2

Theory

In this chapter we cover the theory of the machine learning methods used in this thesis. The chapter is structured as follows. First we describe ways of representing text documents using statistical methods and recurrent neural networks. Thereafter we explain the multi-layer perceptron classifier used in our experiments. Lastly we cover various techniques for training a model as well as tackling the problem of training a classifier on an imbalanced dataset.

2.1 Representation

For classification tasks, text documents of variable length need to be transformed into a representation that a classifier is able to comprehend, typically a fixed-size vector representation. In this section we focus on a document representation that is learned for the task at hand using recurrent neural networks. However, we begin with describing some static document representations constructed using statistical methods and feature engineering, namely bag-of-words representation, n -gram representation and topic models, since these have been reported to be successful in the Fake News Challenge stage 1 competition.

2.1.1 Text Documents

The bag-of-words model represents a text document by its multiset of words. A multiset stores all occurrences of an element but ignores order. This means that the bag-of-words model does not capture any spatial information, such as word-word co-occurrence in a text document. Instead of considering each word in a text document as element, an n -gram model captures spatial information by storing the occurrences of n words appearing in sequence in the document.

For document classification tasks, the bag-of-words or the n -gram model is mainly used for extracting features from a text document into a fixed-size vector representation [16]. The size of the document representation is determined by the size of the vocabulary used for a corpus (a large collection of text documents). A commonly used feature is the term frequency, $\text{TF}_{t,d}$, counting the occurrences of a term t in a document d in the corpus. However, term frequency weighs all terms equally and does not consider how important a certain term is to a certain document in the corpus. Instead it is common to reduce the term frequency weight by the inverse document frequency. The document frequency, DF_t , is defined by the number of doc-

uments in the corpus that contains a term t . Now the inverse document frequency, IDF_t , is defined by

$$IDF_t = \log\left(\frac{N}{DF_t}\right),$$

where N is the number of documents in the corpus. Normalizing the term frequency weight, $TF_{t,d}$, with the inverse document frequency, IDF_t , is called TF-IDF weighting, $TF\text{-}IDF = TF_{t,d} \times IDF_t$. The TF-IDF weight tends to reduce the impact of words with little discriminative power.

Another useful feature indicating similarity between two documents is the cosine similarity between their normalized bag-of-words term frequency (BoW-TF) vector representations [16]. The cosine similarity between two vectors \mathbf{u} and \mathbf{v} is described by the following equation.

$$\text{cosine}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|}$$

A drawback with calculating document distance using BoW-TF vectors and cosine similarity, is that the BoW-TF vectors are frequently near-orthogonal which means that the cosine similarity is often close to zero. Another drawback is that the distance between individual words is not captured in the BoW-TF vectors. The problem occurs when two documents convey the same information and have few or no words in common [31].

Topic modelling using latent semantic analysis (LSA) is a method that attempts to circumvent the issues mentioned above [31]. This method makes use of the TF-IDF matrix of a corpus to produce a set of concepts or latent topics for analyzing the documents in the corpus in a low-dimensional semantic space [18]. Transforming a corpus into a TF-IDF matrix, each row corresponds to a document in the corpus and the columns represent the TF-IDF weights for unique words. The TF-IDF matrix is a sparse representation of the corpus. LSA aims to identify patterns in the relationships between terms and concepts in such collection using a matrix factorization technique called singular-value decomposition [45]. Applying this technique reduces the number of columns in the sparse TF-IDF matrix, resulting in a more coherent document representation [31]. In the reduced matrix, columns represent latent topics and a row corresponds to the mixture of latent topics in a document. Now the similarity between two documents can be calculated using cosine similarity between their mixtures of topics.

Transforming a document into a fixed-size vector representation using an n -gram model or a topic model results in a simplified representation of the document from which it is easy to extract various features. However, these simplified representations might lose important information disregarding the underlying dynamics of a document. Instead there are methods transforming a text document into a fixed-size vector representation as it is, considering all the information it carries. That is, encoding text represented as a sequence of tokens where a token is a character, number, punctuation, space or other symbol. The recurrent neural network is an

example of encoders operating on sequential input commonly used for natural language processing tasks.

A sequence of tokens representing a text document is typically large, therefore it is common to group the tokens in a document into a sequence of words. It is further possible to reason about a text document in terms of sentences and paragraphs, where a sentence is a sequence of words terminated with a punctuation token. A paragraph groups one or more sentences related to the same topic. The document is now constructed out of a sequence of paragraphs delimited with space. Tokens and words in a document reside in a discrete space, where a token maps to an index in an alphabet and a word maps to an index in a vocabulary. However, when encoding a text document for document classification tasks, it is common to represent words in a continuous space as vectors embedding linguistic information about them (word vector embeddings), see Section 2.1.2. An important advantage with word vector embeddings for document classification tasks, is that they can capture how similar two words are. Recurrent neural networks can be used to encode the dynamic representation of a text document described here into a fixed-size vector representation capturing contextual information, see Section 2.2.

2.1.2 Words

A unique word in a corpus maps to an index in a defined vocabulary. This index can be used to represent the word for natural language processing tasks. However, for computational reasons, e.g. processing text using recurrent neural networks, it is preferable to represent words with vectors. One way is to use discrete one-hot vectors. A one-hot vector is a vector where a single element is one and the rest of the elements are zero. The index of the element set to one in the one-hot vector maps to the word in the vocabulary at that index. However, this means that the dimension of the one-hot representation increases with the size of the vocabulary used. Thus, in a large corpus the one-hot encoding of words tend to become an excessively large and sparse representation.

Another way is to represent words using dense vectors in a relatively low-dimensional space embedding some linguistic information about the words, so called word vector embeddings. Word embeddings are constructed based on the distributional hypothesis stating that words used in the same contexts tend to have similar meanings [24]. There are two main model families for constructing word embeddings, namely methods based on global matrix factorization, e.g. LSA, and local context window methods, e.g. the models in word2vec [34]. Word2vec is a group of models utilizing either the continuous bag-of-words model architecture or the continuous skip-gram model architecture, where the former predicts the current word given the context window of surrounding words while the latter predicts the context window of surrounding words given the current word [33]. GloVe is another unsupervised learning algorithm for constructing word embeddings utilizing the word-word co-occurrence statistics in a corpus. The GloVe algorithm combines the advantages of global matrix factorization methods and local context window methods [34].

The aforementioned methods produce a continuous word vector space taking a large corpus as input. Words that are used in the same contexts are close in the resulting vector space and the semantic similarity between two words can be measured by the cosine similarity between their word vectors. Also, the authors showed that vector operations on word2vec or GloVe word vector embeddings often preserve semantic relationships [33] [34], for example the vector difference $v(\text{King}) - v(\text{Man})$ is roughly equal to $v(\text{Royalty})$.

2.2 Recurrent Neural Networks

The recurrent neural network (RNN) is a family of neural networks characterized by a graph of computational units forming a feedback loop [22]. This network topology allows to process sequences of arbitrary length and to hold an internal state between time steps. The internal state can be seen as a memory aiming to capture temporal dynamics in the input [12]. When processing a text document, the internal state of the RNN allows holding information about the whole document. The basic RNN unit is described by the recursive function

$$\mathbf{h}_t = f\left(\mathbf{W}^h \mathbf{x}_t + \mathbf{U}^h \mathbf{h}_{t-1} + \mathbf{b}^h\right),$$

where f is an activation function, \mathbf{x}_t is the input at time step t , \mathbf{h}_{t-1} is the hidden state calculated at the previous time step and \mathbf{h}_t is the new hidden state. The parameters of the RNN unit, weight matrices \mathbf{W}^h and \mathbf{U}^h and bias vector \mathbf{b}^h , are learned using a supervised training procedure referred to as backpropagation through time, see Section 2.4. Given an initial state \mathbf{h}_0 , the output from an RNN processing an input sequence, $\mathbf{x}_1, \dots, \mathbf{x}_T$, is a sequence of states, $\mathbf{h}_1, \dots, \mathbf{h}_T$. If the input is a text document represented as a sequence of word vector embeddings, the output is a sequence of contextual vector embeddings and the last contextual embedding, \mathbf{h}_T , represents the context of the entire document.

2.2.1 Long Short-Term Memory

A problem with the aforementioned RNN architecture is that it might suffer from vanishing or exploding gradients during training when backpropagating the errors through each time step of a sequence. If the gradients tend to vanish it might be problematic for the RNN to capture long-term dependencies in sequences and, conversely, if the gradients tend to explode it might be difficult training the network due to oscillating weights [26]. However, there are several variants of the RNN architecture aiming to circumvent the problem of vanishing or exploding gradients. One of these variants uses so called long short-term memory (LSTM) cells in its internal state initially proposed by Hochreiter and Schmidhuber [27]. The internal state of an RNN with LSTM architecture is much more complex than in the basic RNN unit. The idea of an LSTM unit is that it has a memory content and gates that regulates how much of the input is added to this and if to forget current memory content. Furthermore, the LSTM unit has gates regulating how much memory

content is exposed to the output. The LSTM unit used in this thesis is described by the following recursive functions

$$\begin{aligned}\mathbf{i}_t &= \sigma(\mathbf{W}^i \mathbf{x}_t + \mathbf{U}^i \mathbf{h}_{t-1} + \mathbf{b}^i), \\ \mathbf{f}_t &= \sigma(\mathbf{W}^f \mathbf{x}_t + \mathbf{U}^f \mathbf{h}_{t-1} + \mathbf{b}^f), \\ \mathbf{o}_t &= \sigma(\mathbf{W}^o \mathbf{x}_t + \mathbf{U}^o \mathbf{h}_{t-1} + \mathbf{b}^o), \\ \mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}^c \mathbf{x}_t + \mathbf{U}^c \mathbf{h}_{t-1} + \mathbf{b}^c), \text{ and} \\ \mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t),\end{aligned}$$

where \odot is the elementwise multiplication, σ is the sigmoid function and \tanh is short for the hyperbolic tangent function. The memory of the LSTM unit at time step t is represented by \mathbf{c}_t and the output is \mathbf{h}_t . The gates in the LSTM unit, the input gates \mathbf{i}_t , the forget gates \mathbf{f}_t , and the output gates \mathbf{o}_t , take values in the range of 0 to 1 representing how much information they let through at time step t . The LSTM architecture is parameterized with weight matrices \mathbf{W}^* and \mathbf{U}^* and bias vectors \mathbf{b}^* .

2.2.2 Gated Recurrent Units

The gated recurrent unit (GRU) is another RNN architecture, similar to the LSTM unit, aiming to circumvent the problem of vanishing or exploding gradients [15]. The GRU architecture is less complex than the LSTM architecture and they are comparable in performance on many tasks [17] [28] [11]. The main difference is that GRU has no memory state, instead it exposes the hidden state without regulation. The GRU architecture is described by the following recursive functions

$$\begin{aligned}\mathbf{z}_t &= \sigma(\mathbf{W}^z \mathbf{x}_t + \mathbf{U}^z \mathbf{s}_{t-1} + \mathbf{b}^z), \\ \mathbf{r}_t &= \sigma(\mathbf{W}^r \mathbf{x}_t + \mathbf{U}^r \mathbf{s}_{t-1} + \mathbf{b}^r), \\ \mathbf{h}_t &= \tanh(\mathbf{W}^h \mathbf{x}_t + \mathbf{U}^h (\mathbf{s}_{t-1} \odot \mathbf{r}_t) + \mathbf{b}^h), \text{ and} \\ \mathbf{s}_t &= (1 - \mathbf{z}_t) \odot \mathbf{h}_t + \mathbf{z}_t \odot \mathbf{s}_{t-1},\end{aligned}$$

where \odot is the elementwise multiplication, σ is the sigmoid function and \tanh is short for the hyperbolic tangent function. The reset gates, \mathbf{r}_t , regulate how much information in the previous state, \mathbf{s}_{t-1} , is added to the hidden state, \mathbf{h}_t , and the update gates, \mathbf{z}_t , regulate what information to keep from the hidden state and the previous state constituting the new state, \mathbf{s}_t . The GRU architecture is parameterized with weight matrices \mathbf{W}^* and \mathbf{U}^* and bias vectors \mathbf{b}^* .

2.2.3 Bidirectional Units

Recurrent neural networks can also be constructed bidirectionally as first proposed by Schuster and Paliwal [38]. In a bidirectional RNN, the input is processed both forwards and backwards simultaneously. To accomplish this, each unit holds two states, one for positive time direction and one for negative time direction. The two states in each unit are not interconnected such that the state output from one direction does not influence the other direction. This means that at any given time step

t , the information available in the network consists of information from the current time step, \mathbf{x}_t , information from all previous time steps of the input, $\mathbf{x}_0, \dots, \mathbf{x}_{t-1}$, and information from all future time steps, $\mathbf{x}_{t+1}, \dots, \mathbf{x}_T$. The bidirectional RNN is known to improve the representation of a sequence [23].

2.3 Multi-Layer Perceptron

A multi-layer perceptron (MLP), also called feedforward neural network, stacks several layers of nodes: an input layer, an output layer and one or more intermediate layers referred to as hidden layers [22]. Nodes in the hidden layer(s) and the output layer are computational units applying a non-linear activation function to a weighted input. A hidden layer allows the MLP to distinguish data that is not linearly separable. The MLP is for this reason commonly used as a classifier, mapping a set of input features propagated through each hidden layer to the nodes in the output layer representing different classes. The softmax output layer is used for multi-class classification tasks,

$$\mathcal{S}(\mathbf{x})_c = \frac{e^{\mathbf{w}_c^T \mathbf{x}}}{\sum_{c'=1}^C e^{\mathbf{w}_{c'}^T \mathbf{x}}},$$

where C is the set of classes in the classification problem, \mathbf{w}_c is a vector of weights to be learned for class $c \in C$ and \mathbf{x} is an input feature vector from the preceding layer. In a softmax output layer, the output values of all nodes represent a probability distribution over all classes.

An MLP is fully connected, i.e. all nodes in the hidden layer(s) and the output layer are connected by weights to all nodes in the preceding layer. The size of a hidden layer in an MLP refers to the number of nodes it consists of. If the size of a hidden layer is n and the input to the hidden layer is of size m , then the weight matrix \mathbf{W} connecting the nodes of the hidden layer and the preceding layer is of size $m \times n$. The hidden layer can be described with the following equation

$$\mathbf{y} = f(\mathbf{W}^T \mathbf{x} + \mathbf{b}),$$

where $\mathbf{y} \in \mathcal{R}^n$ is the output of the hidden layer, $\mathbf{x} \in \mathcal{R}^m$ is the input to the layer and f is an activation function. The layer is parameterized with the weight matrix $\mathbf{W} \in \mathcal{R}^{m \times n}$ and the bias vector $\mathbf{b} \in \mathcal{R}^n$. The bias is a threshold parameter that shifts the activation function independent of the output from the preceding layer. The weights and biases in an MLP are learned using a supervised training procedure referred to as backpropagation, see Section 2.4.

The sigmoid function, $\sigma(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{x}}}$, and the hyperbolic tangent function (\tanh), $\tanh(\mathbf{x}) = \frac{e^{\mathbf{x}}+e^{-\mathbf{x}}}{e^{\mathbf{x}}+e^{-\mathbf{x}}}$, are common activation functions for a hidden layer in an MLP. Both the sigmoid and the tanh functions have an 'S'-shaped form taking values in the ranges of 0 to 1 and -1 to 1 respectively. Note that the sigmoid and the tanh functions can be used in the output layer for binary classification tasks. Another common activation function for a hidden layer is the rectifier defined as $\text{ReLu}(\mathbf{x}) = \max(0, \mathbf{x})$, i.e. a linear function for positive input and zero for negative inputs.

2.4 Training

Training a neural network for classification using supervised learning aims to find the network parameters minimizing the error rate given a labelled dataset [22]. The supervised training procedure infers a function that maps a given input to a class label. The goal is that the learned function can be used for mapping previously unseen inputs well, called generalization, which requires the function to model the underlying relationship in the labelled examples from the training dataset.

2.4.1 Loss Function

Training a neural network on a labelled dataset, the loss function is a function of the difference between the actual labels and the predicted output. The idea is to map the parameters of the neural network onto the loss of making prediction errors and the training procedure seeks to find parameters minimizing the total loss on the labelled dataset. A commonly used loss function for training neural networks on classification tasks is the cross-entropy loss function, which measures how close the output probability distribution of a classifier, \hat{Y}_i , is to the corresponding target distribution, Y_i , for each training example i ,

$$L(\mathbf{Y}, \hat{\mathbf{Y}}) = - \sum_i^N Y_i^T \log \hat{Y}_i.$$

2.4.2 Gradient Descent

Gradient descent is an iterative optimization algorithm minimizing a function by taking small steps in the direction of the negative gradient [22]. The gradient descent algorithm is commonly used for adjusting the parameters training a neural network. Propagating a training example through a neural network, the gradients of the loss function are the error contributions of the parameters mapping the input to a label. A gradient descent iteration calculates the gradients of the loss function and updates the parameters of the network aiming to reduce the errors made on a set of training examples. The algorithm iterates over a labelled dataset guiding the parameters with the goal of converging to a state where the network minimizes the loss function. For stochastic gradient descent, an iteration is a single training example while for mini-batch gradient descent, an iteration is a batch of training examples. Iterating over the entire training dataset is commonly referred to as an epoch. The gradient descent algorithm typically needs several epochs training a neural network before reaching the convergence criteria.

The gradient descent update step of a weight matrix \mathbf{W} in a neural network can be described by

$$\mathbf{W} = \mathbf{W} - \eta \frac{\partial L}{\partial \mathbf{W}},$$

where L is the loss function and η is the learning rate, also called the step size, determining the proportion of the gradients used to update the network parameters.

For a feedforward neural network stacking multiple layers on top of each other, the gradients of parameters in the hidden layers are derived applying the chain rule for each layer. In order to calculate the error contributions of the parameters, the loss calculated at the output of the network must be propagated back through the layers. This technique is also called backpropagation or backward propagation of errors.

Similar to the feedforward neural network topology, a recurrent neural network can be unrolled as a stack of layers where each layer corresponds to a time step of an input sequence. The layers of an unrolled RNN share parameters and applying the chain rule, the error contributions of the parameters are calculated as the sum of the gradients of the loss function over all time steps. This technique is also called backpropagation through time, since the loss must be backpropagated through each time step of the input sequence. The gradient descent update step for a weight matrix \mathbf{W} in an RNN is written as

$$\mathbf{W} = \mathbf{W} - \eta \sum_t \frac{\partial L_t}{\partial \mathbf{W}},$$

where L_t is the loss at time step t .

2.4.3 Generalization

In supervised learning, one speaks of overfitting when the function inferred tends to describe outliers and noise instead of capturing the underlying relationship in the training data. A model that overfits on the training data shows poor predictive performance on unseen examples, implying a generalization error. Overfitting occurs when a model is too complex. For neural networks the problem of overfitting commonly occurs when there are too many parameters relative to the number of training examples and/or when the iterative training procedure is performed too long. Early stopping is commonly used when training a neural network to prevent the model from overfitting on the training dataset. This technique stops the iterative training procedure when the performance on unseen examples starts to decrease.

A regularization scheme adds additional terms to a loss function penalizing complexity in arguments found in the optimization procedure, i.e. hindering the optimization procedure to explore certain regions of a model's parameter space. Early stopping is a form of regularization. The motivation for using regularization is to induce less complex and more sparse models aiming to improve generalization. When training neural networks, L1 and/or L2 regularization terms are commonly added to the loss function such that large weights are penalized. The L1 penalty term is the sum of the absolute value of all weights and the L2 penalty term is the sum of the square of all weights.

Dropout is another commonly used technique for preventing overfitting when training a neural network [40]. The idea is to prevent units from co-adapting by randomly excluding them from the network during training. This is achieved by applying an elementwise multiplication between the activation of a unit and a vector of independent random variables taking the values of 1 with a fixed keep probability p and 0

with dropout probability $1 - p$. Applying dropout is an efficient way of performing model averaging.

2.5 Class Imbalance

In a dataset with an imbalance in class distribution, the number of examples belonging to each class differs significantly. When training a classifier on an imbalanced dataset, care must be taken evaluating its performance on minority classes. This is due to a shifted class distribution in another domain may severely affect the performance of the classifier, depending on what evaluation metric is used, since it tends to be biased towards majority classes in the development domain. In this section we cover various methods for training and evaluating a classifier on an imbalanced dataset to account for shifts in class distribution.

2.5.1 Performance Metrics

A common performance metric used when evaluating the performance of a classifier on a balanced dataset is accuracy, i.e. the proportion of correctly classified instances. However, accuracy is not a suitable metric for evaluating a classifier in a domain with an imbalance in class distribution since it tells little about the classifier’s performance on minority classes when the classifier achieves high accuracy being biased towards majority classes. Instead, when evaluating the performance of a classifier in an imbalanced domain, it is more common to analyze the confusion matrix [2]. Table 2.1 shows a confusion matrix for a binary-class classification problem, reporting four different results: true positives (TP) denoting the number of positive instances that a classifier correctly predicted as positives, false negatives (FN) denoting the number of positive instances that a classifier incorrectly predicted as negatives, true negatives (TN) denoting the number of negative instances that a classifier correctly predicted as negatives and false positives (FP) denoting the number of negative instances that a classifier incorrectly predicted as positives. The positive class in the confusion matrix corresponds to the minority class while the negative class corresponds to the majority class.

	positive prediction	negative prediction
positive class	true positives (TP)	false negatives (FN)
negative class	false positives (FP)	true negatives (TN)

Table 2.1: Confusion matrix for a binary-class classification problem.

Derived from the results reported in a confusion matrix, the F1-score is a performance metric widely used in imbalanced domains defined as

$$\text{F1-score} = 2 \cdot \frac{\textit{precision} \cdot \textit{recall}}{\textit{precision} + \textit{recall}}$$

where

$$\textit{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}},$$

and

$$\textit{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

Precision is a metric describing a classifier’s exactness, i.e. the proportion of instances predicted as positives that are correct, while recall, also called sensitivity, is a metric describing a classifier’s completeness, i.e. the proportion of positive instances that are correctly predicted as positives. The F1-score punishes a classifier that tends to be biased towards the majority class, seeing that precision and recall is equally weighted, in contrast to accuracy which based on the results reported in a confusion matrix is defined as

$$\textit{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}.$$

2.5.2 Balancing the Class Distribution

One approach to generalize on shifts in class distribution is to pre-process the dataset balancing the class distribution [21]. Balancing the dataset reduces a classifier’s bias towards the majority classes. Mainly there are two techniques for balancing the class distribution of a dataset: (i) over-sampling examples of the minority classes randomly replicating examples and (ii) under-sampling the majority classes randomly removing examples. However, over-sampling the minority classes increases the risk of overfitting and a drawback with under-sampling is that we might lose important information.

2.5.3 Cost-Sensitive Learning

Typically the objective in classification is to minimize the mis-classification rate. In this setting, the costs caused by all types of mis-classifications are considered equal. In cost-sensitive classification, it is assumed that different types of mis-classifications cause different costs [9]. A cost-sensitive learning method incorporates different costs for different types of mis-classifications in the learning algorithm of a classifier [21]. The cost-sensitive learning method can be useful for training a classifier on an imbalanced dataset treating a mis-classification of an instance from the minority class as more costly than mis-classifications of instances from the majority classes.

In cost-sensitive learning, the cost caused by mis-classifying a training example is a function of the actual label and the predicted label, represented as a cost matrix, \mathbf{C} , defined as: $\mathbf{C}[i, j] = \text{cost for classifying a training example of label } i \text{ as label } j$ and $\mathbf{C}[i, i] = 0$, where the costs are parameters to the learning algorithm [30]. The

objective of the cost-sensitive learning procedure is not to minimize the error rate but to minimize the expected mis-classification cost:

$$Cost(\mathbf{Y}, \hat{\mathbf{Y}}) = \sum_i^N \mathbf{C}[\text{argmax}(Y_i), \cdot]^T \hat{Y}_i,$$

where Y_i is the actual label represented by a one-hot vector and \hat{Y}_i is the classifier's output probability distribution for training example i . To address the class imbalance problem, the cost matrix can be defined, for instance, based on the class distribution of the training dataset or based on the evaluation metric used.

2.5.4 Ensemble of Classifiers

An ensemble learning method trains a set of classifiers aiming to improve predictive performance combining them all together [21]. This technique is also referred to as model averaging. Below we describe two ensemble learning methods that are useful for improving generalization on shifts in class distribution: bootstrap aggregation (bagging) and boosting.

Bagging is an ensemble learning method reducing the variance of a classifier's predictions. The idea is to train an ensemble of classifiers on different subsets of an original training dataset having different class distributions [21]. The different training sets are sampled uniformly and with replacement from the original training dataset. This means that some training examples may be replicated in the sampled sets. Having an ensemble of classifiers trained on different training sets, the final classification is the class that gets the majority of their individual votes, i.e. the class that the majority of the classifiers predicted.

Boosting algorithms aim to train several weak classifiers combining them into a single strong classifier [21]. The performance of a weak classifier is slightly better than simply guessing labels at random. Each weak classifier is trained on a subset of the training dataset focusing on examples mis-classified by previous classifiers. Having a set of weak classifiers, the boosting algorithm learns a weighted sum of their individual estimates constituting a single strong classifier. This method aims to get more accurate predictions with reduced bias and variance.

3

Fake News Challenge Stage 1

The Fake News Challenge stage 1 (FNC-1) competition was organized by Dean Pomerleau, an adjunct faculty member at Carnegie Mellon University, and Delip Rao, founder of Joostware AI Research Corporation, with the goal of exploring how machine learning and natural language processing can be utilized to combat fake news [35]. The competition aimed to explore this challenge focusing on the task of stance detection. In this chapter we describe the dataset published for the competition, the baseline provided by the FNC-1 organizers and the approaches taken by the participants scoring the highest in the competition. Lastly, we review previous work related to the FNC-1 competition.

3.1 Dataset

Some time before the FNC-1 competition was organized, Ferreira and Vlachos presented a novel dataset called *Emergent* supervising the tasks of fact-checking and stance detection [20]. The dataset was built by Craig Silverman and the Tow Center for Digital Journalism at Columbia University (2015), collecting 300 rumoured claims and 2,595 associated news articles that they labelled with an estimate of their veracity. The set of veracity labels consists of *true*, *false* and *unverified* and allows for studying the task of fact-checking. Further, they summarized each news article into a headline and the resulting headline-article pair was labelled with its stance towards the associated claim. The set of labels for the stance classification task consists of *for*, *against* and *observing*.

The dataset provided for the FNC-1 competition is derived from the Emergent dataset [35]. In the FNC-1 dataset, headlines and articles from the Emergent dataset are paired, randomly and based on their relative stance towards the associated claim, moreover divided into two sets of headline-article pairs: related and not related to each other, where the latter set is labelled *unrelated*. Secondly and more challenging, the set of headline-article pairs that is related to each other is further divided into the three classes *agree*, *disagree* and *discuss* and allows for supervising the task of determining the stance of an article relative to a claim given in the associated headline. Table 3.1 shows an example headline together with excerpts from four different articles of different stance towards the claim in the headline.

The FNC-1 dataset consists of a training set and a test set in which there are 49,972 headline-article pairs in the training set and an additional 25,413 pairs in the

3. Fake News Challenge Stage 1

Headline:

"Robert Plant Ripped up \$800M Led Zeppelin Reunion Contract"

Articles:

"... Led Zeppelin's Robert Plant turned down £500 MILLION to reform supergroup. ..."

"... No, Robert Plant did not rip up an \$800 million deal to get Led Zeppelin back together. ..."

"... Robert Plant reportedly tore up an \$800 million Led Zeppelin reunion deal. ..."

"... Richard Branson's Virgin Galactic is set to launch SpaceShipTwo today. ..."

Labels:

agree

disagree

discuss

unrelated

Table 3.1: Headline and article pairs of different labels. The example is taken verbatim from the FNC-1 official site [35].

test set. The headline-article pairs in the training set are constructed out of 1,689 unique headlines and 1,648 unique articles. There are 894 unique headlines in the test set and 904 unique articles. The class distribution in the training set is 73.1% *unrelated*, 7.4% *agree*, 1.7% *disagree* and 17.8% *discuss*. The class distribution in the test set is 72.2% *unrelated*, 7.5% *agree*, 2.7% *disagree* and 17.6% *discuss*. In addition the organizers of the FNC-1 competition suggest an official hold-out set constructed by a random split (based on a fixed seed) in the set of articles along with their corresponding headlines, such that there are 40,350 headline-article pairs in the training set and 9,622 pairs in the hold-out set.

3.2 Evaluation Metric

In the FNC-1 competition, the score for labelling a class correctly is weighted as follows [35]. A score of 0.25 is credited for labelling the *unrelated* class correctly. If the headline and body text pair is related and it is labelled as any of *agree*, *disagree* or *discuss*, a score of 0.25 is credited and an additional score of 0.75 is credited for labelling the correct relationship class as well. We will refer to this weighted score as the Fake News Challenge score (FNC-score). The rationale for a weighted scoring system was that the related versus unrelated classification subtask was expected to be easier and considered less relevant for the superordinate goal of detecting fake news than the stance classification subtask. The performance of a classifier on the FNC-1 test set is measured by the relative FNC-score, i.e. the score achieved normalized by the total score of classifying all headline-article pairs in the set correctly.

3.3 Baseline

The organizers of the FNC-1 competition provided a simple baseline classifier that achieves a relative FNC-score of 75.20 on the test set. The baseline consists of a gradient boosting classifier taking as input a set of hand-crafted features extracted from a headline and the associated article. The features are constructed based on counting the number of overlapping n -grams in a headline-article pair and counting certain words indicating refutation and polarity. See Appendix A.1 for the list of refuting words considered. Below we list the hand-crafted features extracted from a

headline-article pair.

- Counting words considered as refuting appearing in the headline
- Calculating the polarity of the headline and the article based on the number of refuting words appearing in them
- The fraction of overlapping words in the headline-article pair
- Counting the number of times an n -gram of the headline appears in the associated article, where $n \in \{1, 2, 3\}$
- Counting the number of times an n -gram of the headline appears in the introductory paragraph of the associated article, where $n \in \{1, 2, 3\}$

3.4 State-of-the-Art

There was a total of 50 participants in the FNC-1 competition. In Table 3.2 the top 10 teams in the competition are listed.

Team	% Relative FNC-score
SOLAT in the SWEN	82.02
Athene	81.97
UCL Machine Reading	81.72
Chips Ahoy!	80.21
CLUlings	79.73
unconscious bias	79.69
OSU	79.65
MITBusters	79.58
DFKI LT	79.56
GTRI - ICL	79.33

Table 3.2: Top 10 results in the FNC-1 competition

The winner of the FNC-1 competition, team Solat in the SWEN, used a weighted average between a deep convolutional model and gradient boosted decision trees achieving a relative FNC-score of 82.02 on the test set. The deep convolutional model stacked several 1D convolutional layers taking the headline and the associated article as input represented by word2vec word vector embeddings pre-trained on the Google News dataset [22]. The output of the convolutional neural network was further sent to an MLP with three layers. The input to the gradient boosted decision trees consisted of five sets of features extracted from headlines and articles, namely *counting* features, *TF-IDF* features, *SVD* features, *word2vec* features and *sentiment* features. The *counting* features are constructed based on several statistics of overlapping n -grams in a headline-article pair. The *SVD* features are headlines

and articles represented as a mixture of the latent topics found in the FNC-1 corpus [18]. Also the cosine similarity between the *SVD* features are calculated which gives an indication if a headline-article pair is related or not. The *word2vec* features are constructed by taking the sum of the Google News word2vec word vector embeddings of all words in a headline and an article [33]. Lastly, the *sentiment* features are based on separate sentiment scores of a headline and the corresponding article for comparison of their attitudes.

Team Athene ended up on second place in the FNC-1 competition using an ensemble of five separate MLPs, all of them stacking seven hidden layers, where predictions were made based on voting. The MLPs were fed with seven sets of features based on the FNC-1 baseline features, topic models and features for stance classification as suggested by Ferreira and Vlachos [20]. For constructing the topic model features they used non-negative matrix factorization, latent dirichlet allocation and latent semantic analysis [18].

The UCL Machine Reading group ended up on third place in the FNC-1 competition using a single MLP with two layers, stacking a ReLu-layer and a Softmax-layer [36]. The MLP was fed with a single set of features consisting of BoW-TF feature vectors and the cosine similarity between a pair of TF-IDF vectors of a headline and the associated article. When extracting the BoW-TF features they used a vocabulary of the 5,000 most frequent terms in the training set excluding a pre-compiled list of stop words and the TF-IDF matrix was constructed using a vocabulary of the 5,000 most frequent terms in both the training set and the test set excluding the same set of stop words. See Appendix A.3 for the list of stop words they used.

3.5 Related Work

In this section we first cover some useful features for stance classification proposed by Ferreira and Vlachos using the Emergent dataset. Second, we review the task of recognizing textual entailment which is similar to the tasks addressed in the FNC-1 competition.

3.5.1 Features for Stance Classification

Ferreira and Vlachos approached the stance classification task supervised by the Emergent dataset using a logistic regression classifier [20]. As input to the classifier, they experimented with extracting a number of hand-crafted features from the headlines and the claims. The features they extracted from a headline consisted of a BoW-TF vector representation of the headline, whether a headline ends with a question mark or not and the minimum distance from the root to common refuting or hedging words in a headline’s dependency tree. Furthermore, they extracted several headline-claim features aiming to capture the entailment relationship. One headline-claim feature was constructed using an alignment score function for matching stems and paraphrases between the headline and the claim. Another headline-claim feature they used is based on matching subject-verb-object triplets extracted from a

headline and the associated claim. They also used a feature constructed by multiplying the cosine similarities between Google News pre-trained word2vec word vector embeddings of all words in a headline-claim pair.

3.5.2 Recognizing Textual Entailment

In natural language processing, studying textual entailment concerns a three-fold relationship between a premise and an associated hypothesis [19]: whether (i) reading the premise it is possible to infer that the hypothesis is true (positive textual entailment), (ii) the premise contradicts the hypothesis (negative textual entailment), or (iii) the premise neither entails nor contradicts the hypothesis. Recently, a dataset for the task of recognizing textual entailment (RTE) was published by Bowman et al. [10]. The dataset consists of 570,152 labelled sentence pairs of high quality. Here we summarize the field of the RTE task supervised by this dataset. We focus on approaches making use of LSTM networks, which includes the current state-of-the-art, since these methods are related to our approach for the tasks in the FNC-1 competition.

Bowman et al. approached the RTE task using two LSTM networks encoding the contextual representation of the premise and the hypothesis separately and taking the concatenation of these as input to an MLP classifier [10]. Rocktäschel et al. approached the task applying a neural attention model on top of a conditional encoder [37]. The conditional encoder they used stacks two LSTM encoders in sequence, where the first LSTM encoder takes as input the premise and the second LSTM encoder is initialized with the final state of the premise and takes as input the hypothesis. At the time of publication they reported state-of-the-art results. Wang et al. modified the LSTM unit to incorporate the neural attention mechanism proposed by Rocktäschel et al. which they call matching LSTM or mLSTM for short [43]. The idea is that the mLSTM unit allows the attention mechanism to remember certain matching results for the long-term as well as to forget less important results. Similar to Bowman et al., they encoded the contextual embeddings of the premise and the hypothesis separately using two LSTM networks. Wang et al. achieved a higher accuracy than Rocktäschel et al.

Sha et al. incorporated an attention mechanism in the LSTM unit based on the contextual embeddings of the premise, the current memory state and the previous attended contextual embedding [39]. This pipeline of processing the premise and the hypothesis is similar to the conditional encoder proposed by Rocktäschel et al. Cheng et al. modified the LSTM unit to have a set of memory states instead of a single memory state. They further employed an attention mechanism in the LSTM unit that for each word of the hypothesis focus on a subset of the memory states important for the word currently being encoded [14]. Another work by Chen et al. modified the LSTM unit such that two sentences are encoded at the same time [13].

Wang et al. tackled the task of RTE amongst other sentence matching tasks proposing a bilateral multi-perspective matching operation [44]. They used two sepa-

3. Fake News Challenge Stage 1

rate LSTM networks as a contextual representation layer and the bilateral multi-perspective matching operation was applied on the contextual embeddings of the premise and the hypothesis. The idea was to weight the contextual embeddings using a perspective weight matrix and then match the different perspectives of the premise and the hypothesis using cosine similarity. They improved on the state-of-the-art results earlier reported by Chen et al.

4

Models

In this chapter we describe the models we investigate for representing news articles and the relationship between a headline and the corresponding body text. Here we present how these models are used for representing the relationship between two natural language sentences. Throughout this chapter we let $(\mathbf{x}_1^s, \dots, \mathbf{x}_N^s)$ denote a sequence of word vectors representing a sentence s of length N .

4.1 Parallel Encoder

Bowman et al. demonstrated that RNNs can learn efficient semantic representations of natural language sentences for the task of recognizing textual entailment [10] [37]. They encoded a representation of two sentences separately using two LSTMs and the concatenation of these were used as input to an MLP classifier. We will refer to this model as the parallel encoder. Even more successful approaches using the parallel encoder architecture additionally employed an intermediate attention layer before the classifier, e.g. the mLSTM network proposed by Wang et al. [43], or a matching layer, e.g. the multi-perspective matching layer proposed by Wang et al. which is the current state-of-the-art [44].

The parallel encoder processes two sentences, s and t , as follows:

$$\begin{array}{ll} \mathbf{h}_1^s = \text{RNN}^s(\mathbf{x}_1^s, \mathbf{h}_0) & \mathbf{h}_1^t = \text{RNN}^t(\mathbf{x}_1^t, \mathbf{h}_0) \\ \vdots & \vdots \\ \mathbf{h}_M^s = \text{RNN}^s(\mathbf{x}_M^s, \mathbf{h}_{M-1}^s) & \mathbf{h}_N^t = \text{RNN}^t(\mathbf{x}_N^t, \mathbf{h}_{N-1}^t) \end{array}$$

where \mathbf{h}_0 is an initial state of zeros. The final states, \mathbf{h}_M^s and \mathbf{h}_N^t , correspond to the semantic representations of the two sentences. The parallel encoder is illustrated in Figure 4.1.

4.2 Conditional Encoder

Previous work reports the conditional encoder architecture being successful for the task of recognizing the entailment relationship between two natural language sentences [37]. A recent work by Augenstein et al. used a bidirectional conditional encoder for the task of stance classification in tweets [4]. The tweets express a positive or negative attitude against a certain target. An example is the target

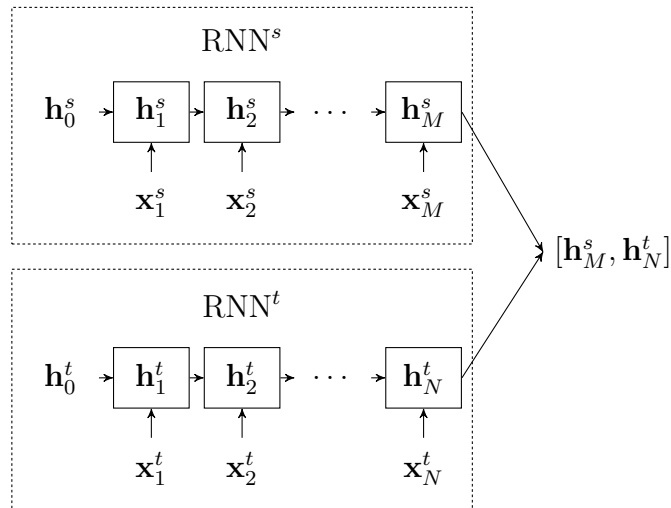


Figure 4.1: Illustration of the parallel encoder. Nodes with superscript s and t represent a sentence s and another sentence t respectively. The final states of the two RNNs, \mathbf{h}_M^s and \mathbf{h}_N^t , correspond to the semantic representations of the two sentences. The concatenation of the final states is denoted $[\mathbf{h}_M^s, \mathbf{h}_N^t]$.

"Legalization of Abortion" and the corresponding tweet "A foetus has rights too!". Augenstein et al. achieved second best performance on the SemEval 2016 Task 6 Twitter Stance Detection corpus.

In contrast to determining entailment of two sentences reading them independently reasoning over their semantic representations, the conditional encoder aims to model the entailment relationship reading both sentences in one go reasoning over combinations of words and phrases [37]. As illustrated in Figure 4.2, the conditional encoder employs two RNNs in sequence, where the first RNN takes as input a sentence s and the second RNN takes as input another sentence t initialized with the representation of the first sentence:

$$\begin{aligned}
 \mathbf{h}_1^s &= \text{RNN}^s(\mathbf{x}_1^s, \mathbf{h}_0) \\
 &\vdots \\
 \mathbf{h}_M^s &= \text{RNN}^s(\mathbf{x}_M^s, \mathbf{h}_{M-1}^s) \\
 \mathbf{h}_1^t &= \text{RNN}^t(\mathbf{x}_1^t, \mathbf{h}_M^s) \\
 &\vdots \\
 \mathbf{h}_N^t &= \text{RNN}^t(\mathbf{x}_N^t, \mathbf{h}_{N-1}^t),
 \end{aligned}$$

where \mathbf{h}_0 is an initial state of zeros [4]. The final hidden state of the second RNN, \mathbf{h}_N^t , corresponds to a representation of the entailment relationship between the two sentences.

The conditional encoder proposed in previous work consists of RNNs with LSTM units [37] [4]. In this setup, the memory state of the second LSTM is initialized with the final memory state of the first LSTM and the hidden state of the second LSTM

is initialized with zeros:

$$\begin{aligned}
 [\mathbf{h}_1^s, \mathbf{c}_1^s] &= \text{LSTM}^s(\mathbf{x}_1^s, \mathbf{h}_0, \mathbf{c}_0) \\
 &\vdots \\
 [\mathbf{h}_M^s, \mathbf{c}_M^s] &= \text{LSTM}^s(\mathbf{x}_M^s, \mathbf{h}_{M-1}^s, \mathbf{c}_{M-1}^s) \\
 [\mathbf{h}_1^t, \mathbf{c}_1^t] &= \text{LSTM}^t(\mathbf{x}_1^t, \mathbf{h}_0, \mathbf{c}_M^s) \\
 &\vdots \\
 [\mathbf{h}_N^t, \mathbf{c}_N^t] &= \text{LSTM}^t(\mathbf{x}_N^t, \mathbf{h}_{N-1}^t, \mathbf{c}_{N-1}^t),
 \end{aligned}$$

where \mathbf{h}_0 and \mathbf{c}_0 are initial states of zeros.

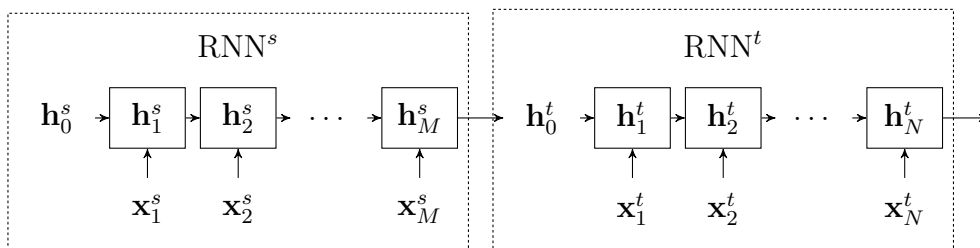


Figure 4.2: Illustration of the conditional encoder. Nodes with superscript s and t represent a sentence s and another sentence t respectively. The final state of the second RNN, \mathbf{h}_N^t , represents the entailment relationship between the two sentences.

4.3 Neural Attention

Neural attention models gained traction when they were introduced in the field of neural machine translation using an encoder-decoder RNN architecture. In this setup, the encoder aims to capture the important features of a sentence to be translated into a fixed-size vector representation. The decoder is initialized with the representation of the source sentence and its task is to generate the translated sentence. However, for a large input space the fixed-size vector representation of a source sentence might impose a bottleneck in the model. To circumvent this bottleneck and improve the performance of the encoder-decoder architecture, the neural attention mechanism proposed by Bahdanau et al. allows the decoder to search through the source sentence for features that are relevant for generating the next word [6].

Rocktäschel et al. approached the task of recognizing textual entailment using the conditional encoder applying a neural attention model on top of the resulting hidden states processing two sentences. Similar to the neural attention mechanism for machine translation, the idea is to circumvent the bottleneck in the conditional encoder architecture initializing the second encoder with a fixed-size vector representation of the first sentence. The attention model employed on the conditional encoder attends to past hidden states of the first sentence such that the RNN encoder does not need to capture the whole semantics of the sentence. Instead, it suffices to inform the second RNN encoder which parts of the first sentence are important such that the

attention model can search for these [37].

As Wang et al. highlight, the main idea with the neural attention model proposed by Rocktäschel et al. is to determine an attention-weighted combination of the hidden states of the first sentence, relevant to each word in the second sentence [43]. That is, for a certain word \mathbf{x}_k^t in the second sentence, an attention vector \mathbf{a}_k aims to model the relevant parts in the first sentence. Wang et al. present the neural attention model as follows. Encoding a sentence t conditioned on another sentence s using the conditional encoder, let \mathbf{h}_k^t and \mathbf{h}_j^s denote the resulting hidden states processing the words \mathbf{x}_k^t and \mathbf{x}_j^s respectively. Now, the aforementioned attention vector \mathbf{a}_k is determined by the following weighted sum:

$$\mathbf{a}_k = \sum_{j=1}^M \alpha_{kj} \mathbf{h}_j^s.$$

Here is α_{kj} an attention weight, representing the degree of alignment between \mathbf{x}_j^s and \mathbf{x}_k^t , generated as follows:

$$\alpha_{kj} = \frac{\exp(e_{kj})}{\sum_{j'} \exp(e_{kj'})},$$

where

$$e_{kj} = \mathbf{w}^e \cdot \tanh(\mathbf{W}^t \mathbf{h}_k^t + \mathbf{W}^s \mathbf{h}_j^s + \mathbf{W}^a \mathbf{h}_{k-1}^a),$$

where \cdot is the dot product, \mathbf{w}^e is a learned vector of weights and \mathbf{W}^* are learned weight matrices. Further, \mathbf{h}_{k-1}^a is a hidden state in an RNN model over all attention weight vectors, $\{\mathbf{a}_k\}_{k=1}^N$, allowing the attention model to keep track of what was attended to in previous steps. The hidden states of the recurrent attention model are defined by:

$$\mathbf{h}_k^a = \mathbf{a}_k + \tanh(\mathbf{W}^a \mathbf{h}_{k-1}^a),$$

where \mathbf{W}^a is a learned weight matrix. The final hidden state of the recurrent attention model, \mathbf{h}_N^a , corresponds to the attention-weighted representation of the first sentence. Rocktäschel et al. used the concatenation of \mathbf{h}_N^a and the final hidden state of the second sentence, \mathbf{h}_N^t , to determine the entailment relationship.

5

Experiments and Results

In this chapter we describe the experiments we have conducted along with the results obtained. The models we present here were trained on the FNC-1 dataset and for all experiments we report the best performing model on the official hold-out set as well as on the test set using the relative FNC-score as evaluation metric. Note that a model evaluated on the hold-out set is not necessarily the same model evaluated on the test set, an issue which is discussed in Section 6.5.

5.1 Representation and Classifier

A news article in the FNC-1 dataset consists of a headline and an associated article. A headline, H , is typically a short text that we invariably represented as a sequence of word vectors. Headlines that consist of more than one sentence were concatenated into a single sentence. An article, A , almost always consists of several sentences and in a set of experiments we investigated representing an article either as a sequence of word vectors or as a set of sentences, where each sentence is a sequence of word vectors. In addition we investigated various attentive representations of articles based on the content in a headline. The experiments we conducted evaluate different models based on the parallel encoder and the conditional encoder, while also comparing LSTM architecture and GRU architecture. The results for the different models described here are reported in Table 5.1, where the parallel encoder is denoted PE and the conditional encoder is denoted CE .

A representation extracted from a news article was used as input to an MLP classifier with two layers. The last layer of the MLP was a softmax layer with an output node for each label in the set of labels $\{unrelated, agree, disagree, discuss\}$. For the hidden layer in the MLP, we compared the efficiency between applying the tanh activation function and the ReLu activation function. The results applying different activation functions are reported in Table 5.1. For all experiments, the size of the hidden layer in the MLP was set to 300 and the size of the encoders were set the same as the dimension of the word vectors used.

Words were represented using 300-dimensional GloVe word vector embeddings pre-trained on the Wikipedia 2014 dataset and the Gigaword 5 dataset [34]. We introduced a token for out of vocabulary words, a token for padding sentences to equal length and a token for representing the end of a sentence. The word vectors for the additional tokens were initialized with zeros. The pre-processing pipeline of

headlines and articles removed punctuation tokens, words appearing only once and words considered stop words. We used the list of stop words proposed by the UCL Machine Reading group [36], see Appendix A.3. We did not optimize word vectors during training.

When training the classifier we minimized the cross entropy of the training set using stochastic gradient descent and the Adam optimization algorithm¹. The initial learning rate was set to be 0.0005 and the first momentum and the second momentum was set to be 0.9 and 0.999 respectively.

In all experiments reported here we used the same random seed to sample the initial weights of the models while all biases were initialized with zeros. We verified the stability of the initial state of the models by running the training procedure of the best setups in the first set of experiments using other random seeds observing no significant deviation in performance on the hold-out set nor on the test set.

Setup	Hold-Out	Test
PE, LSTM, sequence of words, tanh	74.98	57.18
CE, LSTM, $A \rightarrow H$, sequence of words, tanh	74.86	60.60
CE, GRU, $A \rightarrow H$, sequence of words, tanh	80.31	62.61
CE, GRU, $A \rightarrow H$, sequence of words, ReLu	82.83	67.49
CE, GRU, $H \rightarrow A$, sequence of words, ReLu	84.73	64.28
CE, GRU, $\overset{\leftrightarrow}{A} \rightarrow H$, sequence of words, ReLu	86.41	67.84
PE, GRU, sequence of words, ReLu	85.18	66.55
CE, LSTM, $A \rightarrow H$, set of sentences, tanh	85.85	62.83
CE, GRU, $A \rightarrow H$, set of sentences, tanh	86.80	67.51
CE, GRU, $A \rightarrow H$, set of sentences, ReLu	87.30	67.93
CE, GRU, $H \rightarrow A$, set of sentences, ReLu	84.73	64.28
CE, GRU, ReLu, word-by-word attention, sequence of words	83.61	65.57
CE, GRU, ReLu, word-by-word attention, set of sentences	85.88	64.97
CE, GRU, ReLu, sentence attention	87.49	69.08
CE, GRU, ReLu, combined attention	83.65	64.68

Table 5.1: % FNC-score on the hold-out set and the test set for different model setups and activation functions.

¹ Adam is an efficient gradient descent optimization algorithm which works well in practice and is commonly used for natural language processing tasks [29]. The Adam optimizer maintains a per-parameter learning rate improving performance on problems with sparse gradients. Also, the per-parameter learning rates are adapted based on an exponential moving average of the gradients and the squared gradients, increasing performance on noisy problems.

5.1.1 Representing Articles

When representing an article as a sequence of words, we studied the relationship between the headline and the entire article. Using the conditional encoder, we tested both encoding a headline conditioned on the representation of the associated article, $A \rightarrow H$, and the other way around of encoding an article conditioned on the representation of the corresponding headline, $H \rightarrow A$. In addition, we modified the conditional encoder such that the first RNN processes the article bidirectionally into two separate conditional states encoding the headline using two separate RNNs, $\overleftrightarrow{A} \rightarrow H$. The idea behind this relationship representation is that the beginning and the end of an article is more important for detecting the relationship than the content in between.

Due to the frequent asymmetry of an article being a large text document and the associated headline being a short text, we also studied the relationship between each sentence in an article and the headline, representing the article as a set of sentences. In this setup, we made use of the conditional encoder in such way that the headline was encoded conditioned on the representation of each sentence in an article, $A \rightarrow H$, or the other way around such that each sentence was encoded conditioned on the representation of the headline, $H \rightarrow A$. The average of the relationship representations of all sentence-headline pairs represents the relationship between the headline and the entire article. All sentences were padded to equal length allowing them to be processed as a batch of sentences reducing the time for training the model significantly.

5.1.2 Employing Attention

In a set of experiments we investigated whether employing an attention mechanism in the model improves the relationship representation between a headline and the associated article. For these experiments we made use of the conditional encoder, $A \rightarrow H$, and the recurrent neural attention model proposed by Rocktäschel et al [37].

First we investigated the RNN attention model applied on the conditional encoder representing articles as sequences of words, extracting an attention-weighted representation of an article for each word in a headline. The attentive representation of an article corresponding to the last word in a headline was concatenated with the relationship representation extracted using the conditional encoder and then used as input to the classifier. The idea was to incorporate an attention mechanism in the model learning to focus on certain words or phrases in an article that are important for detecting the relationship based on the content in a headline. In another experiment we applied the attention model for each sentence-headline pair. In this setup, the attentive representation of the sentence corresponding to the last word in a headline was concatenated with the sentence-headline relationship representation extracted using the conditional encoder. The resulting set of attentive sentence representations and sentence-headline relationship representations was averaged into a representation of the entire article and its relationship to the associated headline. In Table 5.1 we report the results for these experiments denoted as *word-by-word*

attention.

Another experiment investigated an attention mechanism learning to find relevant sentences in an article based on the content in a headline. In this setup, we extracted a representation of the associated headline using an additional RNN encoder. The final state of this RNN was used as input to the attention model generating attention-weights for the set of sentence representations of an article, i.e. the set of final states of the first RNN in the conditional encoder. These attention-weights were then used for calculating a weighted sum of the set of sentence-headline relationship representations extracted using the conditional encoder. In Table 5.1 we report the results for this experiment denoted as *sentence attention*.

Lastly we conducted an experiment combining the two aforementioned attention mechanisms. More specifically, based on the content in a headline we employed an attention model generating attention-weights for sentences and another attention model generating attention-weights for words in sentences. In Table 5.1 we report the results for this experiment denoted as *combined attention*.

5.2 Learning With Limited Supervision

A relatively large set of training examples is typically needed for a recurrent neural network to properly learn the underlying relationship in the data and perform well on examples it has not previously seen. In the experiments conducted investigating different models, we noticed that these overfit the training data significantly. In particular, we observed in a qualitative analysis of the models that they were sensitive on certain words appearing in headlines or articles. An example of this seen in our analysis was a headline-article pair having no words in common that a model incorrectly believed to be related and simply removing the word "authentic" appearing in the body text made the model believe the pair being unrelated instead. This is demonstrated and discussed further in Chapter 6. Addressing the issue of overfitting the training data, we investigated commonly used methods for reducing generalization errors when training RNNs such as implementing dropout and regularization of parameters. We also investigated a data augmentation method based on resampling examples replacing certain words during training. Another experiment investigated the effects of transfer learning by pre-training an RNN encoder without supervision on another dataset for the task of language modelling.

In the experiments described here, we used the conditional encoder, $A \rightarrow H$, representing articles as sets of sentences since this model was the most efficient to train. Moreover, we used GRU architecture in the RNNs and the ReLu activation function was applied in the hidden layer of the MLP since this setup worked the best in the previous experiments. Since a model has seen all headlines in the official hold-out set during training and seeing that the performance of a model is significantly better on the hold-out set than on the test set, we henceforth put more focus on a model's performance on the test set aiming to reduce the generalization error.

Method	Hold-Out	Test
DW-H	87.34	67.24
DW-A	86.97	67.55
DS $p = 0.5$	87.23	68.99
DS $p = 0.6$	87.59	69.30
DS $p = 0.7$	87.65	67.50
DS $p = 0.5$, Sentence Attention	84.28	62.10
SR-H IDF	87.47	67.30
SR-H DF	87.95	68.88
SR-A DF	86.95	69.40
SR-H DF \times COS	86.34	67.83
SR-A DF \times COS	88.06	69.15
$\lambda_1 = 1e-7$	85.73	67.58
$\lambda_1 = 1e-8$	87.21	68.75
$\lambda_1 = 1e-9$	86.80	68.96
$\lambda_1 = 1e-10$	87.53	68.09
$\lambda_2 = 1e-7$	87.51	67.55
$\lambda_2 = 1e-8$	87.09	68.37
$\lambda_2 = 1e-9$	87.05	68.37
$\lambda_2 = 1e-10$	86.73	68.27
TL, $A \rightarrow H$	82.07	65.92
TL, $H \rightarrow A$	84.07	64.36

Table 5.2: % FNC-score on the hold-out set and the test set investigating different methods to improve the learning algorithm.

5.2.1 Dropout Words

We conducted a set of experiments investigating dropout as regularization method randomly excluding words in headlines or in articles during training. The idea was to train a model such that it generalizes well on words appearing in headlines or articles and learns to distinguish what words in a given context are important for making a certain classification. The dropout strategy we investigated is as follows: For each word in a headline or in an article, keep the word with probability $p = 0.9$. We did not apply this strategy on words we considered being refuting or hedging, see Appendix A.1 and A.2 for these lists of words. In Table 5.2 we report the results for applying dropout on words in headlines and in articles separately denoted as *DW-H* and *DW-A* respectively.

5.2.2 Dropout Sentences

Similar to the idea behind applying dropout on words, we conducted a set of experiments investigating dropout as regularization method randomly excluding sentences in an article during training. The dropout strategy we investigated is as follows: For

each sentence in an article, keep the sentence with probability $p \in \{0.5, 0.6, 0.7\}$. In Table 5.2 we report the results for this set of experiments denoted as *DS*.

5.2.3 Synonym Replacement

In a set of experiments we investigated a data augmentation method based on replacing certain words during training. The goal was to increase the number of training examples artificially improving generalization on words seen in the training set. In the pre-processing stage, we tagged each word in articles and headlines with its part-of-speech tag, one from the set {noun, adjective, verb, adverb}. In this stage, we also extracted synonyms for each word using the WordNet database [42]. Each word was also included in its set of synonyms and the document frequency was calculated for each word in the resulting set. The strategy we investigated for replacing words during training randomly resampled each word into either itself or one of its synonyms using a multinomial distribution. We studied different multinomial distributions for a set of synonyms: (i) normalizing their document frequencies (DF), (ii) normalizing their inverse document frequencies (IDF) and (iii) normalizing their document frequencies multiplied with the cosine similarity between the original word and each synonym ($DF \times \text{COS}$). The difference between (i) and (ii) is that the latter replaces words with synonyms preferring those that are frequently appearing in articles and headlines, while (i) prefers less frequently appearing words. In addition, (iii) prefers replacing words to synonyms that appear frequently and are semantically similar. For replacement we considered only words tagged as adjectives or adverbs. In Table 5.2 we report the results for synonym replacement in headlines and in articles separately denoted as *SR-H* and *SR-A* respectively.

5.2.4 Transfer Learning

Given a model trained for a specific task, transfer learning aims to transfer the knowledge of the given model to improve learning in a new task. In a set of experiments we investigated transfer learning by pre-training the headline RNN in the conditional encoder for a language modelling task. More specifically, we considered the unsupervised task of predicting the next word for each word in a given headline. This task requires the RNN to capture a meaningful semantic representation of the headline. The idea was to get a better initialization of the parameters of the conditional encoder and then fine-tune them on the actual supervised task of detecting the relationship between a headline and the associated article. For this set of experiments, we utilized a combined dataset consisting of the headlines from the FNC-1 training set and headlines from another dataset containing headlines and references to news pages collected from a web aggregator in the period from 10th of March 2014 to 10th of August 2014 [32]. We pre-trained an RNN on the combined dataset for a single epoch and the parameters obtained were used as starting point for the headline RNN in the conditional encoder. The results for these experiments are reported in Table 5.2 denoted as *TL*.

5.2.5 Regularization

Adding L1 and L2 regularization terms to the loss function penalizes large weights and induces sparsity, reducing the complexity of a model. Reducing the complexity of the conditional encoder replacing the LSTM units with GRUs, we noticed a significant improvement in performance on the test set. For this reason we conducted a set of experiments investigating how different values for the L1 regularization multiplier, λ_1 , as well as the L2 regularization multiplier, λ_2 , impacts the performance. In Table 5.2, we report the results for $\lambda_1 \in \{1e-7, 1e-8, 1e-9, 1e-10\}$ and $\lambda_2 \in \{1e-7, 1e-8, 1e-9, 1e-10\}$.

5.3 Addressing the Class Imbalance Problem

In a set of experiments we addressed the class imbalance problem present in the FNC-1 dataset. For this problem we investigated the following methods: balancing the class distribution, cost-sensitive learning and building an ensemble of classifiers. The setup used for this set of experiments was the conditional encoder with GRU architecture, $A \rightarrow H$, representing articles as sets of sentences and the ReLu activation function was applied in the hidden layer of the MLP. The results for the experiments described here are reported in Table 5.3.

Method	Hold-Out	Test
Balancing the Class Distribution	87.46	67.56
$CS = 3.0$	87.21	69.58
$CS = 3.0$, Sentence Attention	87.13	70.05
(iii) $CS = 3.0$, DS $p = 0.5$	87.17	70.23
(ii) $CS = 3.0$, DS $p = 0.6$	86.86	70.29
$CS = 3.0$, SR-A DF	86.09	68.27
$CS = 3.0$, DS $p = 0.5$, SR-A DF	87.65	69.93
$CS = 3.0$, DS $p = 0.5$, SR-H DF	86.66	70.06
$CS = 3.0$, DS $p = 0.6$, $\lambda_1 = 1e-9$	88.01	70.38
$CS = 0.25$	87.10	60.93
$CS = 0.25$, Sentence Attention	86.35	60.02
$CS = 0.25$, DS $p = 0.5$	87.79	60.27
(i) $CS = 0.25$, DS $p = 0.6$	87.25	64.30
$CS = 0.25$, DS $p = 0.6$, $\lambda_1 = 1e-9$	87.89	63.15
Ensemble of Classifiers	-	72.72

Table 5.3: % FNC-score on the hold-out set and the test set addressing the class imbalance problem.

5.3.1 Balancing the Class Distribution

We conducted an experiment balancing the class distribution during training of a classifier, aiming to reduce the classifier’s bias towards the *unrelated* majority class. The strategy we investigated is as follows: For each training example, if the example is labelled *unrelated* include it with probability 0.25 otherwise include the example with probability 1.0. This strategy implies that, (a) the class distribution in the dataset is balanced such that the number of headline-article pairs labelled *unrelated* seen during an epoch is approximately the same as pairs of the stance classification subtask, and (b) the class distribution shifts slightly between epochs.

5.3.2 Cost-Sensitive Learning

In all previous experiments the cost caused by all types of mis-classifications were considered equal, i.e. the mis-classification cost-sensitivity matrix was defined as: $\mathbf{C}[\cdot, \cdot] = 1.0$. In a set of experiments we investigated increasing the cost for headline-article pairs of the stance classification subtask as they are minority in the FNC-1 dataset and since the evaluation criteria favors these, defining the cost-sensitivity matrix as: $\mathbf{C}[\textit{unrelated}, \cdot] = 1.0$ and $\mathbf{C}[i, \cdot] = 3.0$, where $i \in \{\textit{agree}, \textit{disagree}, \textit{discuss}\}$. In addition we conducted a set of experiments reducing the mis-classification cost for headline-article pairs of the stance classification subtask, aiming to increase the classifier’s performance on the *unrelated* class. Since the majority of the headline-article pairs in the FNC-1 dataset are labelled *unrelated* we did not increase the cost for this class. For these experiments we defined the cost-sensitivity matrix as: $\mathbf{C}[\textit{unrelated}, \cdot] = 1.0$ and $\mathbf{C}[i, \cdot] = 0.25$, where $i \in \{\textit{agree}, \textit{disagree}, \textit{discuss}\}$. In Table 5.3, we report the results for the sets of experiments described here denoting their cost-sensitivity settings as $CS = 3.0$ and $CS = 0.25$ respectively.

5.3.3 Ensemble of Classifiers

Lastly we constructed an ensemble of classifiers based on their individual performance on the related versus unrelated classification subtask and the stance classification subtask. The ensemble consisted of three classifiers: (i) a classifier trained reducing the mis-classification cost for headline-article pairs of the stance classification subtask, (ii) a classifier trained increasing the cost for headline-article pairs of the stance classification subtask and (iii) another classifier trained increasing the cost for headline-article pairs of the stance classification subtask but with different regularization setting such that there occurs disagreement between (ii) and (iii) on headline-article pairs labelled *unrelated*. Predictions of the ensemble were made based on the following scheme: For a given headline-article pair, let all classifiers vote for the headline-article pair being unrelated or related and if they agree on a pair being related, then if possible use classifier (ii) to determine the relationship class, otherwise use classifier (iii). The different classifiers that performed the best in the ensemble setup described here are denoted with (i), (ii) and (iii) in Table 5.3.

5.4 Summary

According to the results reported above the ensemble classifier performs the best on the FNC-1 test set. In Tables 5.4, 5.5, 5.6, 5.7, 5.9, 5.8 and 5.10, we report the confusion matrix on the hold-out set and the test set for the ensemble classifier as well as for each individual classifier in the ensemble. For comparison to classifier (i) and classifier (ii) in the ensemble classifier, we also report the confusion matrix on the hold-out set and the test set for the classifier trained with the same setup but considering the cost equal for all types of mis-classifications in Table 5.11 and 5.12. The confusion matrices for the baseline provided by the FNC-1 organizers and the state-of-the-art bag-of-words model used by the UCL Machine Reading group are reported in Table 5.13 and Table 5.14.

Studying the confusion matrices on the test set reported here, we note that the performance on the related versus unrelated classification subtask for the conditional encoder based classifiers is significantly lower than for both the baseline and the state-of-the-art. For the ensemble classifier we report an F1-score of 0.7310 on this subtask, while the baseline and the bag-of-words model score 0.8364 and 0.9374 respectively. Moreover, classifier (i) in the ensemble achieves an F1-score of 0.6486 and classifier (ii) achieves an F1-score of 0.6890, while the classifier trained with the same setup but considering the cost equal for all types of mis-classifications achieves an F1-score of 0.7043. Focusing on the stance classification subtask, we report that the average accuracy on the related set of classes is 45.05% for the ensemble classifier, while 43.98% for the bag-of-words model and 26.23% for the FNC-1 baseline. Moreover, classifier (i) in the ensemble achieves an average accuracy of 27.22% and classifier (ii) achieves an average accuracy of 45.07%, while the classifier trained with the same setup but considering the cost equal for all types of mis-classifications achieves an average accuracy of 36.22%.

	unrelated	agree	disagree	discuss
unrelated	6,668	86	10	134
agree	60	424	10	268
disagree	15	31	67	49
discuss	71	86	41	1,602
FNC-score: 3,881.25 out of 4,448.50 (87.25%)				

Table 5.4: Confusion matrix on the hold-out set for classifier (i) in the ensemble classifier.

	unrelated	agree	disagree	discuss
unrelated	6,171	281	65	381
agree	21	514	9	218
disagree	4	10	96	52
discuss	27	134	45	1,594
FNC-score: 3,863.75 out of 4,448.50 (86.86%)				

Table 5.5: Confusion matrix on the hold-out set for classifier (ii) in the ensemble classifier.

	unrelated	agree	disagree	discuss
unrelated	6,477	173	52	196
agree	25	542	17	178
disagree	14	41	93	14
discuss	47	207	49	1,497
FNC-score: 3,877.75 out of 4,448.50 (87.17%)				

Table 5.6: Confusion matrix on the hold-out set for classifier (iii) in the ensemble classifier.

	unrelated	agree	disagree	discuss
unrelated	16,995	433	9	912
agree	1,033	428	8	434
disagree	476	114	17	90
discuss	1,515	378	38	2,533
FNC-score: 7,492.25 out of 11,651.25 (64.30%)				

Table 5.7: Confusion matrix on the test set for classifier (i) in the ensemble classifier.

	unrelated	agree	disagree	discuss
unrelated	13,423	2,132	160	2,634
agree	212	1,088	30	573
disagree	133	305	42	217
discuss	417	772	61	3,214
FNC-score: 8,189.25 out of 11,651.25 (70.29%)				

Table 5.8: Confusion matrix on the test set for classifier (ii) in the ensemble classifier.

	unrelated	agree	disagree	discuss
unrelated	13,193	2,029	280	2,847
agree	164	968	85	686
disagree	77	293	59	268
discuss	340	726	74	3,324
FNC-score: 8,182.25 out of 11,651.25 (70.23%)				

Table 5.9: Confusion matrix on the test set for classifier (iii) in the ensemble classifier.

	unrelated	agree	disagree	discuss
unrelated	14,553	1,730	100	1,966
agree	256	1,058	33	556
disagree	148	294	46	209
discuss	404	747	57	3,256
FNC-score: 8,472.25 out of 11,651.25 (72.72%)				

Table 5.10: Confusion matrix on the test set for the ensemble classifier.

	unrelated	agree	disagree	discuss
unrelated	6,545	130	27	196
agree	27	569	10	156
disagree	14	49	80	19
discuss	41	229	46	1,484
FNC-score: 3,896.50 out of 4,448.50 (87.59%)				

Table 5.11: Confusion matrix on the hold-out set for the conditional encoder based classifier, $A \rightarrow H$, representing articles as sets of sentences, trained with dropout on sentences in an article, setting the probability of keeping a sentence to 0.6, and considering the cost equal for all types of mis-classifications.

	unrelated	agree	disagree	discuss
unrelated	15,523	1,142	14	1,670
agree	552	737	16	598
disagree	306	207	11	173
discuss	830	566	17	3,051
FNC-score: 8,074 out of 11,651.25 (69.30%)				

Table 5.12: Confusion matrix on the test set for the conditional encoder based classifier, $A \rightarrow H$, representing articles as sets of sentences, trained with dropout on sentences in an article, setting the probability of keeping a sentence to 0.6, and considering the cost equal for all types of mis-classifications.

	unrelated	agree	disagree	discuss
unrelated	17,978	10	3	358
agree	285	173	10	1,435
disagree	238	39	7	413
discuss	1,399	221	7	3,556
FNC-score: 8,761.75 out of 11,651.25 (75.20%)				

Table 5.13: Confusion matrix on the test set for the FNC-1 baseline.

	unrelated	agree	disagree	discuss
unrelated	17,963	53	3	330
agree	114	838	12	939
disagree	116	179	46	356
discuss	262	523	46	3,633
FNC-score: 9,521.50 out of 11,651.25 (81.72%)				

Table 5.14: Confusion matrix on the test set for the state-of-the-art bag-of-words model.

6

Discussion and Analysis

The results reported in this thesis indicate that the RNN based classifiers are weak on the subtask of distinguishing if a headline-article pair is related or not. This was expected to be an easy task on which the naive baseline provided by the FNC-1 organizers performs significantly better and the state-of-the-art bag-of-words model used by the UCL Machine Reading group performs close to perfect. By implementing cost-sensitive learning we were able to reduce a classifier’s bias towards the *unrelated* class and improve the performance on the stance classification subtask. We report an average accuracy on the subtask comparable to the state-of-the-art also outperforming the baseline. However, we note that the average accuracy reported is quite low and in particular low accuracy is reported on headline-article pairs of the *disagree* class.

In the experiments conducted we found that the RNN based models suffer from overfitting; while modelling the training data well, we report poor performance on unseen examples in the test set, as can be seen in Figure 6.3. Related to this observation we noticed that the models tend to be sensitive when certain words appear in headlines or articles, see Section 6.4. Addressing this, we found that methods modifying the learning algorithm to improve generalization have limited effects on a model’s performance on unseen headline-article pairs. Our main conclusion of these experiments is that the RNN based models are too complicated for the task at hand, facing a high variance issue, where the limited size of the FNC-1 dataset, composed out of a relatively few unique headlines and articles, ends up hindering such a model to learn the underlying relationship between words and phrases in an article and the corresponding headline for different classes. The RNNs operate on an unproportionately large feature space compared to the number of training examples; seeing that a headline-article pair typically contains a large set of words, each word equating to a large set of features and where the order in which words appear increases the function space explored in the optimization procedure. In addition, much of the information in an article is considered noise and does not contribute with any useful content for making a certain classification.

In the rest of this chapter we discuss the models we have investigated as well as the methods studied aiming to improve learning in the FNC-1 domain. In addition we provide a qualitative analysis on various issues observed when testing the RNN based models. Lastly we discuss an important issue regarding model selection using the official hold-out set that needs to be addressed in future work.

6.1 Representation and Classifier

When representing an article as a sequence of words we found that the ReLu-layer in the MLP outperformed the tanh-layer, but for the case when the article is represented as a set of sentences none of them works conclusively better. For the former case we argue that the tanh activation function hurts the optimization procedure. Another finding in our experiments is that the GRU models are more efficient than the LSTM variants. We argue this is due to the former type of RNN units reduces the complexity of the models, seeing that there are relatively few training examples in the FNC-1 dataset.

We found that the conditional encoder works better for the tasks in the FNC-1 competition than the parallel encoder. We argue this is due to the former model does not need to extract a rich semantic representation of the headline. Instead, conditioned on the semantic representation of an article the model can focus on certain words or phrases in a headline when reasoning about their relationship. We argue that this also is the reason why reversing the condition is less efficient, i.e. encoding the article conditioned on the semantic representation of the headline. A headline is typically a short text, often a content-dense summary of the corresponding article introducing the subject(s), the topic(s) and the claim(s) argued for or against in the article. Furthermore, the headlines in the FNC-1 dataset are similar to each other. Many of them are constructed artificially from a real headline by changing the order of some words or whole clauses, adding some word that brings no content or replacing words with their synonyms. Articles on the other hand are often large but content-sparse text documents.

The beginning and the end of an article is often directly related to the content in the headline: The introduction often elaborates on the content in the headline and summarizes what is to come in the article while the end is often a conclusion of the argumentation in the article. We found that encoding articles bidirectionally as condition encoding headlines improves the performance slightly. In this setup the beginning and the end of an article are equally close to the final representation and therefore more likely prominent in the resulting state used as condition encoding the headline.

Reducing the input sequence by modelling the degree of alignment comparing headlines against sentences, i.e. representing articles as sets of sentences, yields a diminishing small improvement in performance. Many sentences in an article elaborate on the topic or give some form of background to the event. Some sentences may therefore be completely unrelated to the content in a headline. This might be problematic for the set of sentences model since all sentences in an article are weighted equally reasoning about the relationship. Addressing this, we found that an attention model learning to search articles for relevant sentences based on the content in a headline and weight these accordingly, yields an improvement of 1.15% on the test set. Similarly we believed that an attention model could learn to align words in sentences relevant to words in the headline. However, interestingly such a model

had a negative impact on the performance increasing the tendencies of overfitting the training data. See Section 6.4 for a qualitative analysis on employing attention in a model.

6.2 Learning With Limited Supervision

Implementing dropout randomly removing sentences in an article during training improved the performance comparable to applying an attention model weighting sentences based on the content in a headline. Both methods seem to improve a set of sentences model such that it learns to distinguish what sentences in an article are important for making a certain classification. However, applying dropout on sentences in combination with employing an attention model reduces the performance significantly, i.e. they seem to be exclusive to each other.

The regularization method of dropping words during training affects the performance negatively. Moreover, we found it difficult to artificially augment the set of training examples for the purpose of improving generalization. Synonym replacement seemed promising but combined with other methods the improvements were diminishing. We found replacement of words based on document frequency the most efficient of the replacement strategies, for which including the semantic similarity aspect when resampling words yields no improvement.

Moreover, we found that for some setups adding L1 and L2 regularization terms to the loss function yields an improvement in performance on the test set. However, in combination with other techniques, such as implementing dropout and cost sensitive learning, the improvements were diminishing. Due to this we did not focus on finding an optimal L1 and L2 regularization of the loss function for all different setups.

Interestingly we found that pre-training the headline RNN for language modelling decreases the performance for the setup of encoding the headline conditioned on the representation of an article but increases the performance slightly when encoding the headline and the article the other way around. Related to the discussion regarding representation before, this indicates that the conditional encoder for the setup of encoding the headline conditioned on the representation of an article is not focusing on extracting a meaningful semantic representation of headlines when trained on the FNC-1 dataset.

6.3 Approaching the Class Imbalance Problem

Balancing the class distribution of the dataset decreases the performance on the joint task reducing the FNC-score on the test set. We argue this is due to removing important relationship information seeing that the dataset is constructed combining a relatively small amount of unique headlines and articles.

Compared to considering the cost equal for all types of mis-classifications training a classifier, increasing the mis-classification cost for headline-article pairs labelled as related we found an improvement in performance on the stance classification subtask while the performance was slightly reduced on the related versus unrelated classification subtask; seeing a reduced precision on the union of the related set of classes. Conversely, reducing the mis-classification cost for headline-article pairs labelled as related, we found that the performance on both the stance classification subtask and the related versus unrelated classification subtask was reduced; seeing that a classifier’s recall on the union of the related set of classes was reduced significantly. This observation indicates that cost-sensitive learning reduces or increases a classifier’s bias towards the *unrelated* majority class and allows for selecting the stance classification subtask to focus on. Focusing on the stance classification subtask improves the performance on the joint task increasing the FNC-score on the test set.

Constructing an ensemble combining different models trained with different cost-sensitivity settings, we were able to improve the performance on the related versus unrelated classification subtask and consequently, improve the performance on the joint task achieving an FNC-score 2.5% lower than the FNC-1 baseline. The ensemble classifier’s performance on distinguishing if a headline-article pair is related or not is, however, still significantly lower than the baseline and the state-of-the-art. We believe that investigating ensemble methods further, focusing on the related versus unrelated classification subtask, has potential to improve the performance to the extent of outperforming the baseline. However, due to the complexity of an ensemble model and considering the efficiency of the naive baseline, we did not want to spend further efforts on such an approach.

6.4 Qualitative Analysis

When testing a conditional encoder based classifier on examples from the test set we found following issues: (i) There are many cases when the headline and the article have many words in common and the pair is labelled as related but the classifier predicts the pair to be unrelated, and (ii) the other way around, there are few or no words in common but the classifier predicts the headline-article pair as related. In Figure 6.1 we analyze an example of the latter case.

The sensitivity on certain words appearing in either an article or a headline discussed along with Figure 6.1 is an indication that the RNN based models fail to generalize either extracting a semantic representation of the article or identifying relevant words in the headline. Furthermore, analyzing statistics of mis-classified examples in the test set, the amount of words in an article or in a headline seems not being a source of errors. There are cases when few words in larger articles trigger certain predictions after reading the corresponding headlines. On the other hand, the typically large number of words in articles is arguably a hardship learning meaningful representations of sentences omitting outliers in such, since the function space that the optimization explores is excessively large and there are a relatively few examples in the dataset to learn from. Moreover, as discussed before the headlines in

Saudi national airline may introduce gender segregation on its flights

The US declared the video of Sotloff to be authentic.

Figure 6.1: The conditional encoder based classifier, encoding the headline conditioned on the representation of the article, predicts the headline-article pair shown in this figure as *disagree* while the actual label is *unrelated*. If the word "authentic" is removed from the article, the classifier predicts the pair as *unrelated*. Removing "gender" in the headline keeping all words in the article, makes the classifier believe the pair is *unrelated*. What we see here is that the model generalizes poorly when removing a certain word in a headline or article. This is an indication that the model suffers from overfitting.

the FNC-1 dataset are similar to each other, where in some cases only a few words differ or the order in which the words appear is changed. With limited supervision, this setup may steer the training of a model to trigger a certain classification when certain words or phrases appear in a headline or article indifferent to the context and the conditional state.

Related to issue (i) we found that some errors occur when the headline and the article talk about the same named entities but the topics discussed differ and some errors occur the other way around when the headline and the article discuss the same topics but not the same named entities. An example of this seen in the analysis is a headline talking about "Apple Watches" while the topic of the article is related to "Apple iPhones". This headline-article pair is not related since the topics slightly differ even though the named entity "Apple" is present in both.

Another issue we found, regarding representation of the relationship between a headline and an article, is that several headline-article pairs labelled as *agree* and *disagree* contain few sentences in the article that actually express a positive or negative attitude towards the claim in the headline. As discussed before, many sentences in an article elaborate on the topic or give some form of background to the event. We believe this issue makes it hard for a model to learn separating between examples labelled *agree* and *discuss* as well as between examples labelled *disagree* and *discuss*. We noticed that when the number of sentences in an article increases, the model is more biased towards making a neutral prediction.

6.4.1 Attention Models

The attention model learning to weight sentences in an article relevant to the content in a headline seems to improve the relationship representation of a headline-article pair while the word-by-word attention mechanism seems to fail significantly. To get a better understanding why this is the case, we analyzed attention-weights along with their corresponding sentences and words in several headline-article pairs in the test set.

Man saved from bear attack - thanks to his Justin Bieber ringtone

Justin Bieber may not have been able to take on Orlando Bloom, but he sure as hell was able to take on a bear.

No, PETA, Bieber didn't beat down a bear, he just scared one away in Russia's Yakutia Republic.

It all went down when 42-year-old fisherman Igor Vorozhbitysyn's attack by a brown bear was interrupted by a Justin Bieber ringtone.

The fisherman explains: "I had parked my car and was walking towards the spot I'd marked out when there was a tremendous impact on my back and the bear was on top of me."

That's when Vorozhbitysyn's phone went off and "Baby" started blaring, causing the bear to run off.

He also justified his surprising ringtone: "I know that sort of ringtone isn't to everyone's taste," he said, "but my granddaughter loaded it onto my phone for a joke."

Living with embarrassment is a small price to pay for your life.

Figure 6.2: Illustrating weighted sentences and words in a headline-article pair from the FNC-1 test set, employing an attention model on top of the conditional encoder. The pair is labelled *agree* and the predicted label agrees with the actual label.

Figure 6.2 shows an example headline-article pair where we argue that the sentence attention model successfully pays additional attention to a subset of the sentences in the article. In this type of visualization, a darker blue shade represents a larger attention weight value. Although some sentences are weighted less than others, all sentences are considered making the classification noticing that the attention weight distribution is close to using the average weighting of all sentences. We observed that there is low variance in the weight distribution in almost all headline-article pairs analyzed, i.e. the attention model seems careful attending to certain sentences. In the headline-article pair in Figure 6.2, the sentence "It all went down when 42-year-old fisherman Igor Vorozhbitysyn's attack by brown bear was interrupted by a Justin Bieber ringtone." is weighted the most. This sentence is semantically similar to the headline and also expresses a positive stance towards the headline which agrees with the label of the example. The last sentence, "Living with embarrassment is a small price to pay for your life.", is weighted the least and seeing to the content in the headline it is not notably relevant. However, this sentence subtly indicates that the person who was attacked survived. Related to this observation, we noticed that the sentence attention model can fail when some sentences may seem completely unrelated to the headline but they indeed express an attitude towards the claim in the headline. Instead the attitude towards the headline expressed in

a sentence is dependent on a context given in earlier sentences in the article and thus the attention model might not consider such a sentence. On a different note, the attention model also suffers from the issues discussed before regarding learning meaningful semantic representations of headlines and sentences. This might be a limitation in learning what sentences are relevant.

We argue the word-level attention model operates on too fine granularity. Typically we observed that the word-by-word attention model succeeds in weighting overlapping words between a headline and an article. This can be seen in Figure 6.2, showing what words are weighted after reading the last word in the headline. These attention-weights are used to construct the attentive representations of sentences which are part of the input to the classifier. The representation of articles based on weighting words is not efficient seeing a significant lower performance on the test set. It seems that generalization errors increase when the model tends to focus on certain words instead of the semantics of a sentence captured in the conditional state.

6.4.2 Reducing and Increasing Bias

A bias vector, \mathbf{b} , in the softmax layer of the MLP consists of a threshold value for each class in the problem, i.e. $\mathbf{b} = [b_{unrelated}, b_{agree}, b_{disagree}, b_{discuss}]^T$. Studying the resulting bias vectors in the softmax layer when balancing the class distribution during training and for learning with different cost-sensitivity settings, we found that the threshold for the *unrelated* class is adjustable to some extent. Here we report the resulting bias vector for a classifier trained

- a) balancing the class distribution:

$$[-0.47344029, 0.18956994, 0.0220722, 0.21321553]^T$$

- b) considering the cost equal for all types of mis-classifications:

$$[-0.28969282, 0.10894272, -0.15185457, 0.12703601]^T$$

- c) increasing the mis-classification cost for headline-article pairs of the stance classification subtask:

$$[-0.51706803, 0.10909189, -0.01229266, 0.19181129]^T$$

- d) reducing the mis-classification cost for headline-article pairs of the stance classification subtask:

$$[-0.05494795, 0.03138153, -0.11980191, 0.05739412]^T$$

6.4.3 Difficulties and Limitations

Analyzing articles in the FNC-1 dataset we found usage of metaphores, humour, sarcasm and common phrases. Due to the limited dataset, we argue that the model

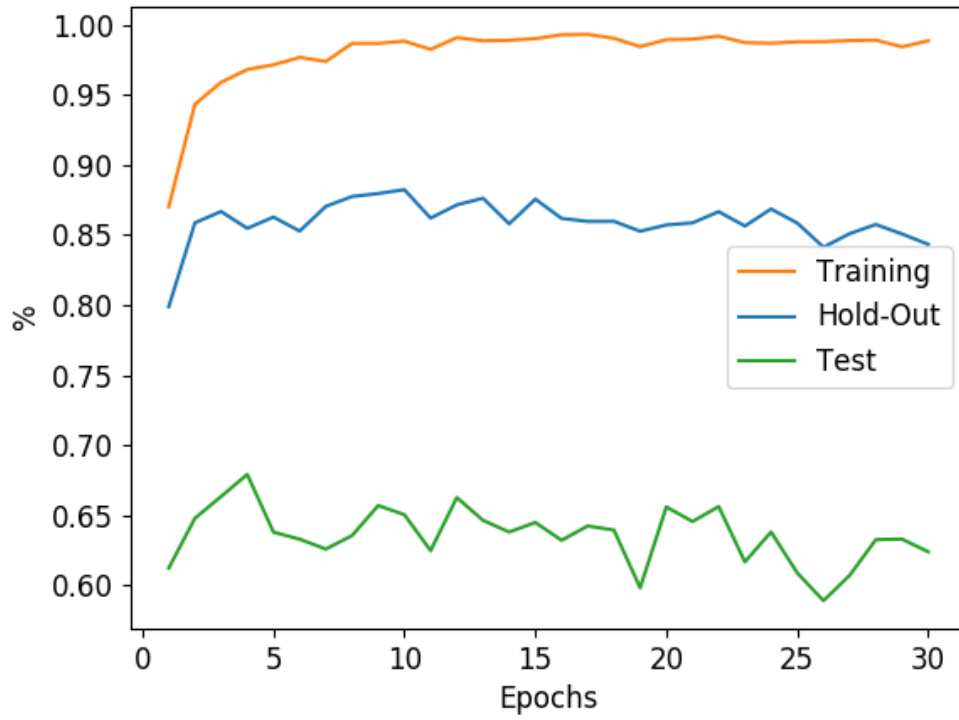
is not able to learn such usage of the language and this is a potential error factor. For instance, a sarcastic sentence typically express a certain attitude towards a subject but the actual meaning is the opposite.

Attending to named entities seems important for the subtask of distinguishing if a headline-article pair is related or not. If both the article and the headline talk about the same entity, the pair is likely related if they also discuss the same topic(s). We found that there are headline-article pairs in the FNC-1 dataset discussing named entities that are not included in the vocabulary. These words are represented with zero vectors and this might have an impact on the performance since the model for such headline-article pair is not able to tell if the headline and the article talk about the same entities. One way to address this problem is to extract word embeddings for missing named entities by averaging over similar entities. For instance, a word embedding for an organization that is not in the vocabulary can be created averaging over similar organizations. In addition, some of the out-of-vocabulary words can be directly mapped to words already present in the vocabulary. For instance, "isis" is in the vocabulary we used but not "isil" and these words are of equal meaning. Thus we could map occurrences of "isil" to "isis" instead. In a simple experiment we did this mapping and noticed a small improvement in performance on the hold-out set as well as the test set. We did not address this issue further as it is not always as obvious as the examples discussed here.

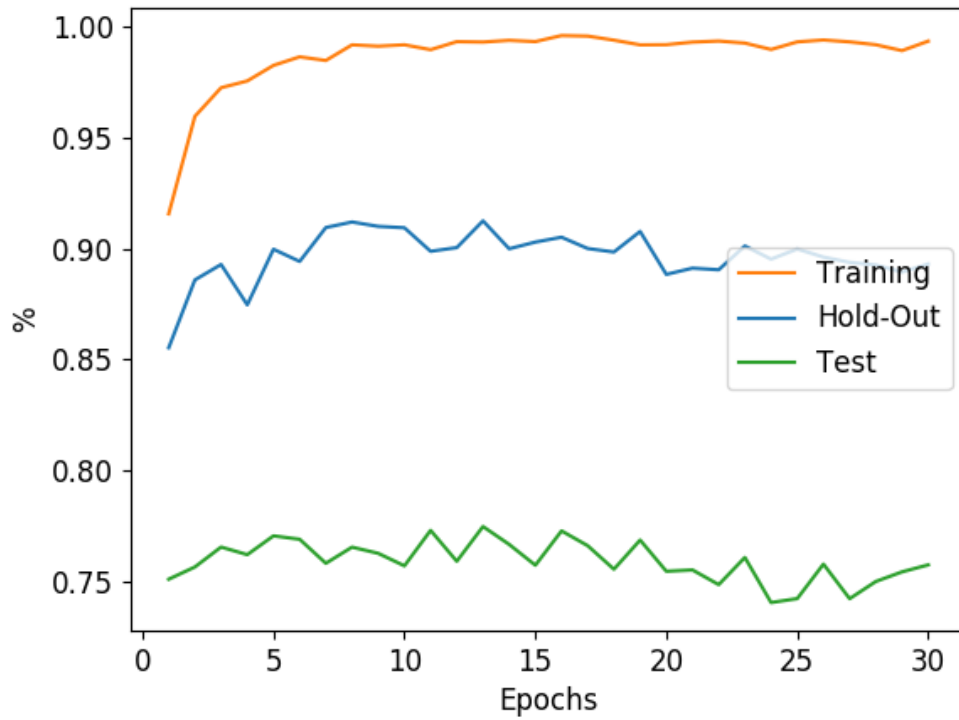
6.5 Model Selection

The FNC-1 training dataset consists of 49,972 headline-article pairs, composed out of 1,689 unique headlines and 1,648 unique articles. The official hold-out set is constructed by a random split in the set of articles along with their corresponding headlines, such that there are 40,350 headline-article pairs in the training set and 9,622 pairs in the hold-out set. Moreover this means that a model evaluated on the official hold-out set has seen all headlines during training. As a consequence we noticed that the RNN based models we trained perform much better on the hold-out set than on the test set. More importantly, we were not able to use the performance of a model on the hold-out set as basis for selecting which model to be evaluated on the test set, since the performance on the hold-out set is not representative for the performance on the test set. This can be seen in Figure 6.3.

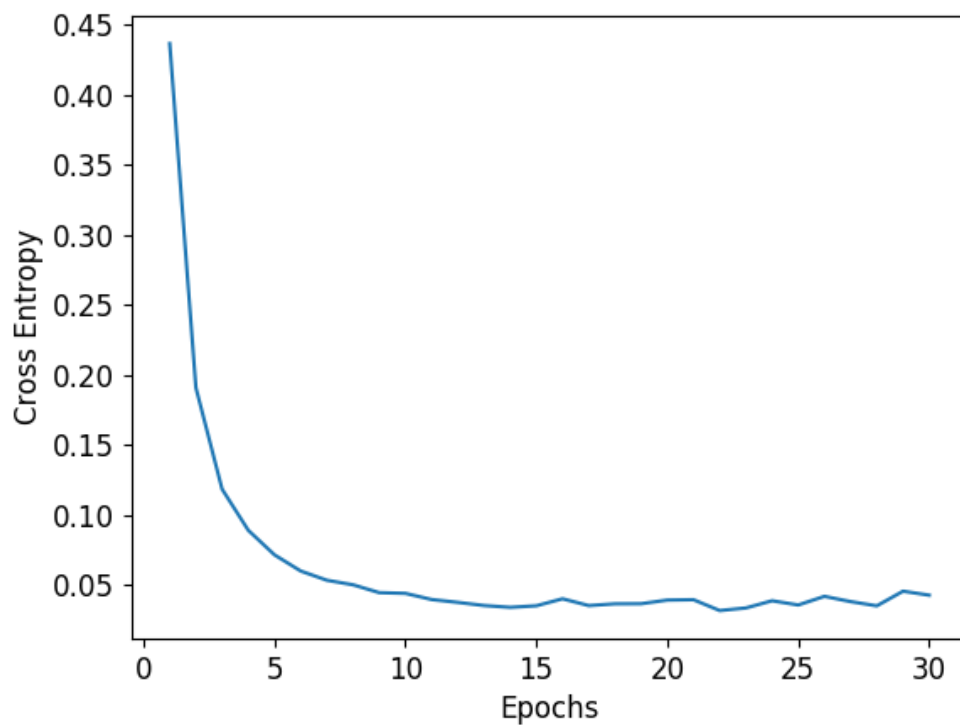
Previous work addressing the issue of headlines bleeding over in the hold-out set suggests a split of the related labels into several disjoint sets evaluating different models using k -fold cross validation [25]. This approach is not suitable for the RNN based models investigated in this thesis since it reduces the amount of training examples significantly. Instead we tested removing headlines overlapping the hold-out set and the training set reducing the training set to 32,444 headline-article pairs and the hold-out set to 2,220 pairs. However, there are still headlines in the non-overlapping hold-out set that are almost identical to headlines in the training set since, as discussed before, many headlines are constructed artificially by altering an original headline slightly. Another issue regarding representing the test data is that



(a) FNC-score



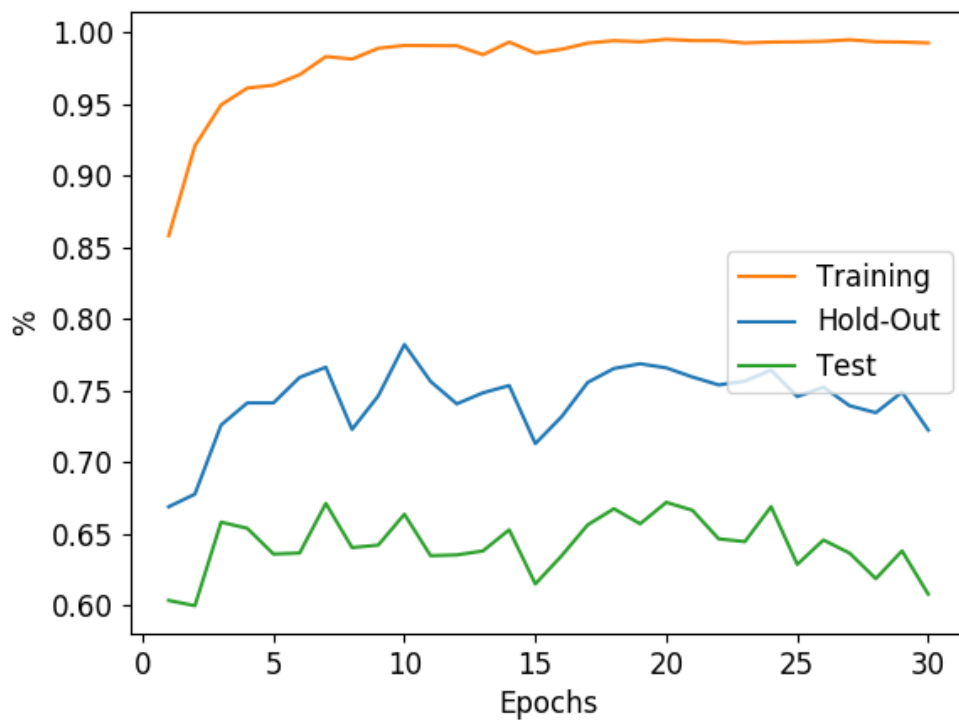
(b) Accuracy



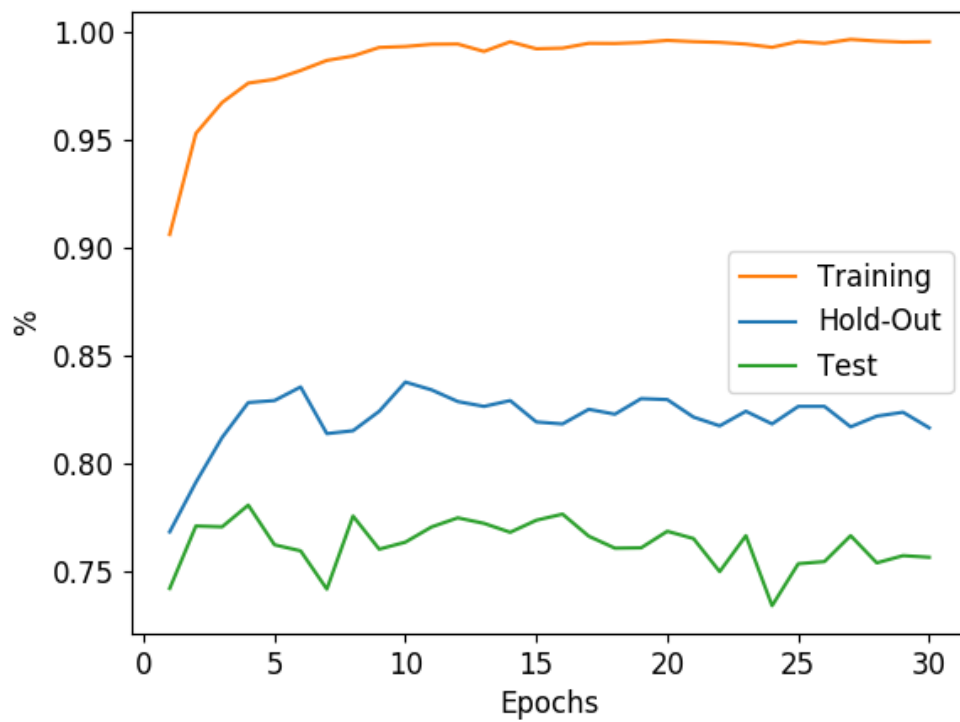
(c) Cross Entropy

Figure 6.3: Plotting the training procedure of an RNN based model for 30 epochs, where (a) and (b) shows the relative fnc-score and the accuracy on the test set, the official hold-out set and the training set for each epoch and (c) shows the cross entropy loss on each training epoch.

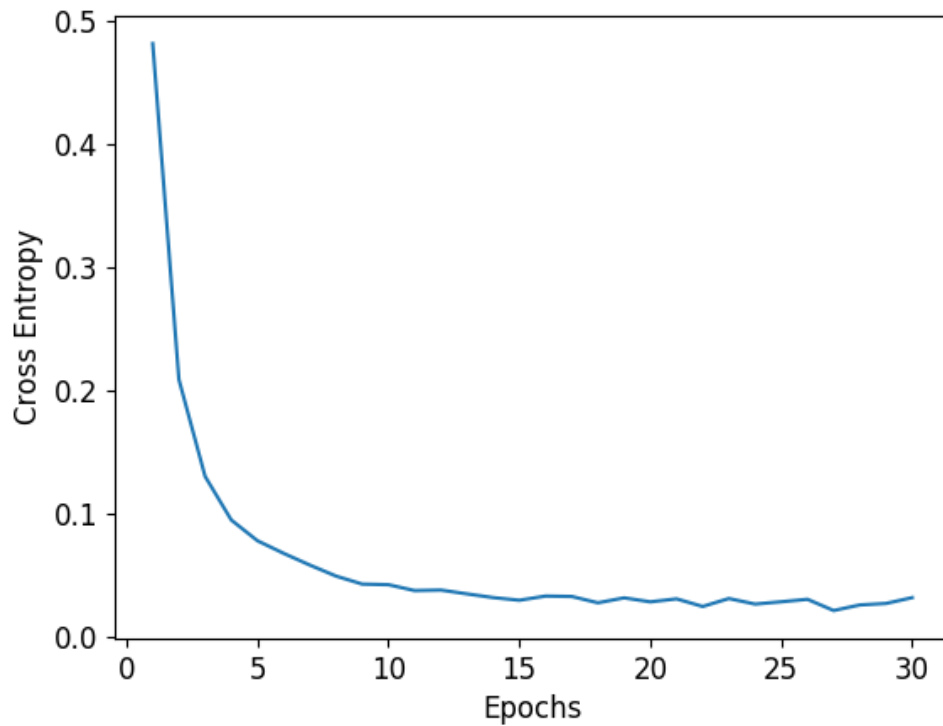
the amount of headline-article pairs in the non-overlapping hold-out set is less than 10% of the training set. Figure 6.4 shows the training procedure using the non-overlapping hold-out and training sets. First we note that the performance on the hold-out set is significantly reduced while the performance on the test set is almost unharmed. Second we note that the metric curves still fluctuate between epochs. However, the fnc-score curve on the test set now tends to follow the fluctuations of the fnc-score curve on the hold-out set. Therefore the non-overlapping hold-out and training sets might be helpful for model selection. We did not investigate this further but the problem needs to be addressed in future work studying RNN based models.



(a) FNC-score



(b) Accuracy



(c) Cross Entropy

Figure 6.4: Plotting the training procedure of an RNN based model for 30 epochs, where (a) and (b) shows the relative fnc-score and the accuracy on the test set, the non-overlapping hold-out and training sets for each epoch and (c) shows the cross entropy loss on each training epoch.

7

Conclusion

In this thesis, we have investigated the effectiveness of representing news articles using recurrent neural networks for the joint task of (i) detecting if a headline is related to an associated body text or not, and, if they are related to each other, (ii) detecting the stance of the body text relative to the headline. For supervision of the task we made use of the dataset delivered for stage 1 of the Fake News Challenge (FNC-1). While the most successful approaches have used complex ensemble classifiers employing a large set of hand-engineered features, their performance is just marginally better than a straightforward bag-of-words model deriving only lexical similarity. Mainly we studied two different RNN based models in this thesis: (i) extracting semantic representations of a headline and body text pair encoding them in parallel, and (ii) extracting an entailment relationship representation encoding a headline conditioned on the semantic representation of a body text, or the other way around. We also studied methods for improving learning with limited supervision as well as addressing the class imbalance problem present in the FNC-1 dataset.

We found that the RNN based models are weak on the subtask of detecting if a headline is related to a body text or not. This was expected to be an easy task on which the FNC-1 baseline performs significantly better and the state-of-the-art systems perform close to perfect. We found that implementing cost-sensitive learning efficiently reduces or increases a classifier’s bias towards the *unrelated* majority class. Reducing the bias, focusing on the stance detection subtask, we found that the conditional RNN models are comparable to the state-of-the-art systems and outperforms the baseline, however, noticing that the average accuracy on the labels of the subtask is quite low.

Our study was severely limited by the amount of training examples in the FNC-1 dataset. The RNN based models are too complex for the task at hand facing a high variance issue that leads to overfitting. The methods studied aiming to improve generalization on unseen headline and body text pairs had a limited effect. However, a significant improvement in performance was seen when reducing the complexity of a model’s architecture switching from LSTM units to GRUs.

Clearly stance detection in the news domain is a difficult task, on which both the RNN based models studied in this thesis and the state-of-the-art systems show disappointing performance. In order to investigate the full potential of the RNN based methods, we would suggest extending the amount of training examples in a dataset for supervising the task.

Bibliography

- [1] M. B. Alexi Mostrous and K. Gibbons. Russia used Twitter bots and trolls ‘to disrupt’ Brexit vote | News | The Times & The Sunday Times. <https://www.thetimes.co.uk/article/russia-used-web-posts-to-disrupt-brexit-vote-h9nv5zg6c>. Visited on 2017-12-03.
- [2] A. Ali, S. M. Shamsuddin, and A. L. Ralescu. Classification with class imbalance problem: A Review. *Int. J. Advance Soft Compu. Appl*, 7(3), 2015.
- [3] H. Allcott and M. Gentzkow. Social Media and Fake News in the 2016 Election. *Journal of Economic Perspectives—Volume*, 31(2—Spring):211–236, 2017.
- [4] I. Augenstein, T. Rocktäschel, and A. Vlachos. Stance detection with bidirectional conditional encoding, 2016.
- [5] M. Babakar and W. Moy. The State of Automated Factchecking How to make factchecking dramatically more effective with technology we have now. https://fullfact.org/media/uploads/full_fact-the_state_of_automated_factchecking_aug_2016.pdf. Visited on 2018-01-09.
- [6] D. Bahdanau, K. Cho, and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate, 2014.
- [7] M. Barthel, A. Mitchell, and J. Holcomb. Many Americans Believe Fake News Is Sowing Confusion | Pew Research Center. <http://www.journalism.org/2016/12/15/many-americans-believe-fake-news-is-sowing-confusion/>. Visited on 2017-03-14.
- [8] N. Bartlett. Russia using ‘fake news to influence key elections’ in France, Germany and the Netherlands - Mirror Online. <http://www.mirror.co.uk/news/world-news/russia-using-fake-news-influence-9685349>. Visited on 2017-10-23.
- [9] B. Bischl, M. Lang, L. Kotthoff, J. Schiffner, J. Richter, E. Studerus, G. Casalicchio, and Z. M. Jones. mlr: Machine learning in r. *Journal of Machine Learning Research*, 17(170):1–5, 2016.
- [10] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference, 2015.
- [11] D. Britz. Recurrent neural network tutorial, part 4 - implementing a gru/lstm rnn with python and theano. <http://www.wildml.com/>, 2015. Visited on 2018-01-09.
- [12] J. A. Bullinaria. Recurrent neural networks. <http://www.cs.bham.ac.uk/~jxb/INC/112.pdf>, 2015. Visited on 2017-04-22.
- [13] Q. Chen, X. Zhu, Z. Ling, S. Wei, H. Jiang, and D. Inkpen. Enhanced LSTM for Natural Language Inference, 2017.

- [14] J. Cheng, L. Dong, and M. Lapata. Long Short-Term Memory-Networks for Machine Reading, 2016.
- [15] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation, 2014.
- [16] P. R. Christopher D. Manning and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.
- [17] J. Chung, C. Gülçehre, K. Cho, and Y. Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, dec 2014.
- [18] S. P. Crain, K. Zhou, S.-H. Yang, and H. Zha. *Dimensionality Reduction and Topic Modeling: From Latent Semantic Indexing to Latent Dirichlet Allocation and Beyond*, pages 129–161. Springer US, Boston, MA, 2012.
- [19] I. Dagan, O. Glickman, and B. Magnini. The PASCAL Recognising Textual Entailment Challenge. In *Machine Learning Challenges. Lecture Notes in Computer Science*, 3944:177–190, 2006.
- [20] W. Ferreira and A. Vlachos. Emergent: a novel data-set for stance classification. *of the 2016 Conference of the North . . .*, 2016.
- [21] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera. A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches, 2011.
- [22] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [23] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 4, pages 2047–2052. IEEE, 2005.
- [24] Z. Harris. Distributional structure. *Word*, 10(23):146–162, 1954.
- [25] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*, chapter 7, pages 241–249. Springer, 2008.
- [26] S. Hochreiter, Y. Bengio, and P. Frasconi. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In J. Kolen and S. Kremer, editors, *Field Guide to Dynamical Recurrent Networks*. IEEE Press, 2001.
- [27] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, nov 1997.
- [28] R. Jozefowicz, W. Zaremba, and I. Sutskever. An empirical exploration of recurrent network architectures. *Journal of Machine Learning Research*, 2015.
- [29] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization, dec 2014.
- [30] M. Kukar and I. Kononenko. Cost-Sensitive Learning with Neural Networks. *13th European Conference on Artificial Intelligence*, 1998.
- [31] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger. From Word Embeddings To Document Distances, 2015.
- [32] M. Lichman. Uci machine learning repository. <http://archive.ics.uci.edu/ml>, 2013. News Aggregator Data Set.
- [33] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.
- [34] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

- [35] D. Pomerleau and D. Rao. Fake news challenge. <http://www.fakenewschallenge.org/>. Visited on 2018-01-09.
- [36] B. Riedel, I. Augenstein, G. P. Spithourakis, and S. Riedel. A simple but tough-to-beat baseline for the Fake News Challenge stance detection task, jul 2017.
- [37] T. Rocktäschel, E. Grefenstette, K. Moritz, H. G. Deepmind, T. Kočisk, and P. Blunsom. Reasoning About Entailment With Neural Attention, 2015.
- [38] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.
- [39] L. Sha, B. Chang, Z. Sui, and S. Li. Reading and Thinking: Re-read LSTM Unit for Textual Entailment Recognition, 2016.
- [40] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.
- [41] S. Tavernise. As Fake News Spreads Lies, More Readers Shrug at the Truth - The New York Times. <https://www.nytimes.com/2016/12/06/us/fake-news-partisan-republican-democrat.html>. visited on 2017-10-22.
- [42] P. University. About wordnet. <http://wordnet.princeton.edu>, 2010. WordNet.
- [43] S. Wang and J. Jiang. Learning Natural Language Inference with LSTM, 2016.
- [44] Z. Wang, W. Hamza, and R. Florian. Bilateral Multi-Perspective Matching for Natural Language Sentences, feb 2017.
- [45] H. Yanai, K. Takeuchi, and Y. Takane. *Singular Value Decomposition (SVD)*, pages 125–149. Springer New York, New York, NY, 2011.
- [46] Y. Yang and A. Nenkova. Combining Lexical and Syntactic Features for Detecting Content-dense Texts in News *, 2017.

A

Appendix 1

A.1 Refuting Words

fake, fraud, hoax, false, deny, denies, refute, not, non, despite, nope, doubt, doubts, bogus, debunk, pranks, retract

A.2 Hedging Words

alleged, allegedly, apparently, appear, appears, claim, claims, could, evidently, largely, likely, mainly, may, might, maybe, mostly, perhaps, presumably, probably, purported, purportedly, reported, reportedly, rumor, rumour, rumors, rumours, rumored, rumoured, says, seem, somewhat, supposedly, unconfirmed

A.3 Stop Words

a, about, above, across, after, afterwards, again, against, all, almost, alone, along, already, also, although, always, am, among, amongst, amount, an, and, another, any, anyhow, anyone, anything, anyway, anywhere, are, around, as, at, back, be, became, because, become, becomes, becoming, been, before, beforehand, behind, being, below, beside, besides, between, beyond, bill, both, bottom, but, by, call, can, co, con, could, cry, de, describe, detail, do, done, down, due, during, each, eg, eight, either, eleven, else, elsewhere, empty, enough, etc, even, ever, every, everyone, everything, everywhere, except, few, fifteen, fifty, fill, find, fire, first, five, for, former, formerly, forty, found, four, from, front, full, further, get, give, go, had, has, have, he, hence, her, here, hereafter, hereby, herein, hereupon, hers, herself, him, himself, his, how, however, hundred, i, ie, if, in, inc, indeed, interest, into, is, it, its, itself, keep, last, latter, latterly, least, less, ltd, made, many, may, me, meanwhile, might, mill, mine, more, moreover, most, mostly, move, much, must, my, myself, name, namely, neither, nevertheless, next, nine, nobody, now, nowhere, of, off, often, on, once, one, only, onto, or, other, others, otherwise, our, ours, ourselves, out, over, own, part, per, perhaps, please, put, rather, re, same, see, serious, several, she, should, show, side, since, sincere, six, sixty, so, some, somehow, someone, something, sometime, sometimes, somewhere, still, such, system, take, ten, than, that, the, their, them, themselves, then, thence, there, thereafter, thereby, therefore, therein, thereupon, these, they, thick, thin, third, this, those, though, three, through, throughout, thru, thus, to, together, too, top, toward, towards, twelve,

A. Appendix 1

twenty, two, un, under, until, up, upon, us, very, via, was, we, well, were, what, whatever, when, whence, whenever, where, whereafter, whereas, whereby, wherein, whereupon, wherever, whether, which, while, whither, who, whoever, whole, whom, whose, why, will, with, within, without, would, yet, you, your, yours, yourself, yourselves