# Combating Fake News with Adversarial Domain Adaptation and Neural Models

by

Brian Xu

B.S., Massachusetts Institute of Technology (2018)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2019



Author .....
Department of Electrical Engineering and Computer Science
February 1, 2019

Certified by .....
James R. Glass
Senior Research Scientist
Thesis Supervisor

Certified by .....
Mitra Mohtarami
Research Scientist
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

# Combating Fake News with Adversarial Domain Adaptation and Neural Models

by

Brian Xu

Submitted to the Department of Electrical Engineering and Computer Science
on February 1, 2019, in partial fulfillment of the
requirements for the degree of
Master of Computer Science and Engineering

## Abstract

Factually incorrect claims on the web and in social media can cause considerable damage to individuals and societies by misleading them. As we enter an era where it is easier than ever to disseminate "fake news" and other dubious claims, automatic fact checking becomes an essential tool to help people discern fact from fiction. In this thesis, we focus on two main tasks: *fact checking* which involves classifying an input claim with respect to its veracity, and *stance detection* which involves determining the perspective of a document with respect to a claim. For the fact checking task, we present Bidirectional Long Short Term Memory (Bi-LSTM) and Convolutional Neural Network (CNN) based models and conduct our experiments on the LIAR dataset [Wang, 2017], a recently released fact checking task. Our model outperforms the state of the art baseline on this dataset. For the stance detection task, we present bag of words (BOW) and CNN based models in *hierarchy* schemes. These architectures are then supplemented with an adversarial domain adaptation technique, which helps the models overcome dataset size limitations. We test the performance of these models by using the Fake News Challenge (FNC) [Pomerleau and Rao, 2017], the Fact Extraction and VERification (FEVER) [Thorne et al., 2018], and the Stanford Natural Language Inference (SNLI) [Bowman et al., 2015] datasets. Our experiments yielded a model which has state of the art performance on FNC target data by using FEVER source data coupled with adversarial domain adaptation [Xu et al., 2018].

Thesis Supervisor: James R. Glass
Title: Senior Research Scientist

Thesis Supervisor: Mitra Mohtarami
Title: Research Scientist

# Acknowledgments

I would like to thank my thesis advisors Mitra Mohtarami and James Glass for their efforts in guiding and assisting with my research. Without their efforts, my thesis would not have been possible. I would also like to thank all of my friends and family for their unwavering love and support throughout my academic journey.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation

The problem of false or misleading news articles, sometimes referred to as "fake news", is not a new phenomena. Even more than a century ago, fake news, under the alias "yellow journalism", had already been exerting massive influence on important global events, such as U.S. foreign policy.[1] Prior to the internet, though, fake news could be contained fairly well because the only effective method of effectively spreading written information was through the newspaper. Almost all newspaper articles were fairly well vetted as they were generally written by professional journalists and peer reviewed by professional editors. Publishing fake news was strongly discouraged, as such an article could easily cost a journalist or editor their job and their reputation.[2]

Now, the rapid rise of social media and other internet media allows anyone to write anything about anything, often in an anonymous manner. Since publishing fake news via this medium is much easier and without significant risk, fake news has seen a resurgence which continues to this day. This rise has prompted increasing awareness of the negative influence of fake news and how it can unfairly influence public opinion on various events and policies [Mihaylov et al., 2015, Mihaylov and Nakov, 2016, Vosoughi et al., 2018]. Today, the fake news problem has, without a

[1] https://history.state.gov/milestones/1866-1898/yellow-journalism
[2] http://time.com/4858683/fact-checking-history/

| Label | Claim |
|---|---|
| pants-fire | In the case of a catastrophic event, the Atlanta-area offices of the Centers for Disease Control and Prevention will self-destruct. |
| half-true | However, it took $19.5 million in Oregon Lottery funds for the Port of Newport to eventually land the new NOAA Marine Operations Center-Pacific. |
| true | The Chicago Bears have had more starting quarterbacks in the last 10 years than the total number of tenured (UW) faculty fired during the last two decades. |

Table 1.1: Examples of data used for the fact checking task.

doubt, made its way into the public limelight, most notably during the recent 2016 US Presidential Election.[3]

Currently, all credible fact checking has to be done manually and the process is very labor intensive. Given a claim, a human fact checker needs to look up different credible sources and discern their stances on the issue. Then, the fact checker must decide, based on that information, whether the claim is true or not. Due to the sheer amount of fact check worthy claims that are posted to the internet every day, it is simply not feasible to have human fact checkers confirm or refute each claim. In order to prevent fake news from misleading readers, a fast, automated mechanism to check the veracity of news claims is necessary. Figuring out how to create such an model is a field of active research which this thesis contributes to. In the next section, we discuss the specific research problems covered in this work.

## 1.2 Problem Descriptions

First, we work on the fact checking problem by framing it as a single classification problem. In particular, given a claim $c$ we aim to determine how factual $c$ is on some predetermined scale of veracity. For this task, we work mainly on the LIAR dataset [Wang, 2017]. This dataset provides a large number of claims and factuality

[3] `https://www.bbc.com/news/world-us-canada-37896753`

| **Claim** | Robert Plant Ripped up $800M Led Zeppelin Reunion Contract. |
|---|---|
| **Label** | **Document** |
| agree | ...Led Zeppelin's Robert Plant turned down £500 MILLION to reform supergroup. ... |
| disagree | ...Robert Plant's publicist has described as "rubbish" a Daily Mirror report that he rejected a £500m Led Zeppelin reunion. ... |
| discuss | ...Robert Plant reportedly tore up an $800 million Led Zeppelin reunion deal. ... |
| unrelated | ...Richard Branson's Virgin Galactic is set to launch SpaceShipTwo today. ... |

Table 1.2: Examples of data used for the stance detection task.

labels scraped from Politifact[4], a reputable fact checking media source. The factuality labels provided for each claim range from *true*, to *barely-true*, to *pants-fire*. Each of the provided claims further comes with some associated metadata, such as the speaker and the subject of the claim. Some examples of LIAR data are displayed in Table 1.1.

Next, we also work on a problem known as the stance detection task which is a critical subtask for commonly proposed automatic fact checking processes. The fact checking problem is often postulated to consist of a multistage process involving the following steps [Vlachos and Riedel, 2014]:

1. Retrieve potentially relevant documents as evidence for the claim [Mihaylova et al., 2018, Karadzhov et al., 2017b].

2. Predict the stance of each document with respect to the claim [Mohtarami et al., 2018, Baly et al., 2018].

3. Estimate the trustworthiness of the documents (e.g. in the Web context, the site of a Web document could be an important indicator of its trustworthiness).

4. Finally, make a decision based on the aggregation of (2) and (3) for all documents from (1) [Mihaylova et al., 2018].

[4] `https://www.politifact.com/`

Stance detection is step (2) of this process. Formally, given a claim $c$ and a document $d$, we aim to determine the stance of $d$ relative to $c$. Stances can be defined one of the following: *agree*, *disagree*, *discuss*, *unrelated*, where a stance of *agree* indicates that a given document $d$ agrees with the corresponding claim $c$. An example for each label is shown in Table 1.2. In this thesis, we present neural based models to address this task and evaluate the models on a publicly available dataset, the Fake News Challenge [Pomerleau and Rao, 2017]. The dataset contains a large number of (claim, document) pairs along with the stance label relating each pair. We also conduct experiments which use the Fact Extraction and VERification dataset [Thorne et al., 2018] and Stanford Natural Language Inference dataset [Bowman et al., 2015] for this task.

## 1.3 Contributions

Our contributions to the previously discussed research problems can be summarized as:

1. We construct neural based models for the fact checking task that can outperform existing baselines for the LIAR dataset.

2. We are the first to apply adversarial domain adaptation for the stance detection task. In particular, we evaluate the relationships between the Fake News Challenge dataset, the Fact Extraction and VERification dataset, and the Stanford Natural Language Inference dataset for this task through adversarial domain adaptation.

3. By leveraging adversarial domain adaptation, we create a stance detection model that can outperform existing state of the art models for the Fake News Challenge dataset.

Each of these contributions will be discussed in detail in the remainder of this thesis. The next section provides an outline of the following chapters.

## 1.4 Outline

Next in Chapter 2, we will discuss previous work related to the fact checking and stance detection problems. Then, in Chapter 3, we will detail our experiments on the fact checking problem evaluated on the LIAR dataset. After, in Chapter 4, we will describe our experiments where we apply adversarial domain adaptation on models for the stance detection task. Following that, in Chapter 5, we explain the design and implementation of a demonstration system for our state of the art stance detection model. Finally, in Chapter 6, we summarize our work and discuss possibilities for future work.

# Chapter 2

# Related Work

In this chapter, we will examine previous work for the fact checking problem and the stance detection problem. For both of these problems, we will survey popular datasets and discuss notable models and results. Also, we will discuss work related to adversarial domain adaptation, a technique we use for the stance detection task. In particular, we will examine its relationship to other domain adaptation methods and notable usages.

## 2.1 Fact Checking

Much of the existing research for the fact checking problem chooses to apply neural methods [Wang, 2017, Karadzhov et al., 2017c, Karadzhov et al., 2017a] such as convolutional neural networks (CNNs) and long short term memory (LSTMs) to solve the problem. In general, text input is converted into high dimensional word embedding vectors, such as pretrained Word2Vec [Mikolov et al., 2013a] or GloVe [Pennington et al., 2014] vectors, before being consumed by those architectures. The word embedding assigned to each word represents that word's meaning in relation to other words.

A large amount of previous work also focuses on extracting a wide variety of other features to help models predict labels. For example, one paper [Karadzhov et al., 2017a] extracts a great variety of features from training data including TF-

IDF features [Ramos et al., 2003], part of speech frequencies and pointwise mutual information (PMI) [Bouma, 2009]. After these features are extracted, classifiers such as a support vector machine (SVM) [Cortes and Vapnik, 1995] can then be used on those features to produce an overall classification.

Other research in the task has focused on extracting information from sources outside the provided training data [Karadzhov et al., 2017c, Ciampaglia et al., 2015]. This is a logical step, because in order for a person to determine the veracity of a claim, he or she would likely start establishing a ground truth by looking at other credible sources. One approach to acquiring useful outside information is through internet search engines. In [Karadzhov et al., 2017c], the proposed model utilizes several search engines including Google, Bing, etc. and retrieves a relevant snippet from each relative to the claim being analyzed. In order to gauge the credibility of the search results found, they compiled a domain blacklist by manually examining the domains of the top 100 most frequently used domains. If a search result comes from a domain on the blacklist, information from that result will not be used. While this filtering is by no means foolproof, as credible domains can still have bad information, authors believe this method provided sufficient information for the task. After applying a series of LSTMs on the representations of the retrieved information, the model is able to use SVMs and multilayer perceptrons to predict the veracity of the original claim.

Another approach taken to solve the fake news problem is to extract features from existing knowledge graphs such as DBpedia [Bizer et al., 2009]. These knowledge graphs are constructed in an attempt to relate the information on the web; DBpedia in particular uses data from Wikimedia and contains information on more than 4.5 million structured content. After defining the concept of semantic proximity and tying a truthfulness metric to the knowledge graph, one paper [Ciampaglia et al., 2015] leverages DBpedia by creating a model which finds the maximal semantic proximity of a claim in question. Based on the truthfulness measure corresponding to the relevant maximal semantic proximity path, the model can then predict whether the claim is true or false. While this approach seems effective for fairly simple fact checking statements (such as “x is married to y”), they noted that the model struggled with

more complex claims.

## 2.2 Stance Detection

The stance detection task is currently a very active area of research. There have been several datasets created for the purpose of stance detection including the Fake News Challenge [Pomerleau and Rao, 2017], SemEval-2016 Task 6 [Mohammad et al., 2016], Emergent [Ferreira and Vlachos, 2016], and most recently the Fact Extraction and VERification dataset [Thorne et al., 2018]. A variety of models have been used to achieve top performing results on these datasets.

One common approach which achieved a top result on the FNC dataset is the multilayer perceptron model [Riedel et al., 2017]. In this model, TF and TF-IDF features are used to represent the input. After passing the input vector through a hidden layer, the model predicts the input article's stance using a softmax layer. Despite the simplicity of this model, it yields one of the top results for the task.

Next, the best performing model for the FNC dataset effectively applied a combination of a decision tree model and a deep CNN model [Baird et al., 2017]. The decision tree model involves first extracting many features from the given claim and document, including count features, sentiment features, etc. Then, a gradient boosted decision tree model is applied on these features to produce a classification. The CNN model is implemented by applying convolutions to the word embedding representation of the data, a common text classification technique [Kim, 2014]. The outputs of the convolutional layers are then fed to a series of fully connected layers before being fed into a final softmax layer for classification. The logits produced from these two models are then combined to make a final prediction.

Another relatively simple but effective method for stance detection is a LSTM model approach. This method was used by MITRE [Zarrella and Marsh, 2016] and produced the top score on SemEval-2016 Task 6 Task A. After converting the text input into a matrix of word vectors, their model fed that matrix into a LSTM model (one word at a time) along with some metadata on the subject of the input text. The

final LSTM output was then fed into a hidden layer which was then used to predict the stance label of the input text. Their results demonstrate the predictive power of LSTMs on short to medium length text inputs.

The bidirectional variant of the LSTM architecture, in which the input text is passed both forwards and in reverse, has also been successfully used to achieve state of the art results on the same dataset [Augenstein et al., 2016]. The model first processes the subject of a claim through a bidirectional LSTM. Then, the output is fed in as metadata to a separate bidirectional LSTM which processes the actual claim. In this way, the model is able to to pay specific attention to subjects.

Finally, the recently proposed memory network architecture [Sukhbaatar et al., 2015] has been successfully used for the stance detection task [Mohtarami et al., 2018]. Memory networks were designed to remember past information, and have been shown to perform better than traditional architectures such as RNNs in many applications. This is because memory networks are better suited to handle semantic relationships with entire paragraphs of text, which can be useful for tasks such as stance detection.

## 2.3 Adversarial Domain Adaptation

The purpose of domain adaptation is to configure data that is useful in the *source* domain for use in a different *target* domain. This technique is especially useful when the *source* domain has plentiful labeled data, but the *target* domain has little to no labeled data. When using data from a different domain, it is usually necessary modify it to fit the *target* domain as there are usually differences in data distributions. There have been a large variety of ideas on how to conduct domain adaptation; we will highlight several of these approaches next. This problem is still an area of active research.

Of the many possible ways to conduct domain adaptation, we are particularly interested in the approaches which require minimal dataset specific alterations. One possible approach to accomplish this is to sample data from a carefully selected source domain which is similar to the target domain [Gong et al., 2013]. Known as land-

marks, these similar pieces of data can be automatically identified. After identification, landmarks can then be used to effectively relate the source domain and the target domain, allowing for provably easier domain adaptation tasks.

Another approach is to conduct a feature space transformation of the source data so that it is closer to that of the target data by using a method such as transfer component analysis [Pan et al., 2011]. The transfer component analysis method attempts to find transfer components across domains in a Reproducing Kernel Hilbert Space using Minimum Mean Discrepancy to measure of distance between distributions. The transfer components are selected so that the subspace spanned by these components contains data from different domains which are similar in distribution to each other. So, in theory, data from one domain from this space can then be used in a different domain without concerns about distribution related discrepancies.

Adversarial domain adaption differs from previous approaches in that it is able to directly minimize the difference in feature space distributions [Ganin and Lempitsky, 2014]. This method works by training an adversary that tries to predict, given the features produced by the model, the domain from which the features came from. If the adversary can accurately predict the original domain, then it was able to identify distribution related differences in the features produced from the different domains. Because the model tries to maximize the adversary's loss, the model is incentivized to learn to choose features which ensure that the adversary performs poorly. So, the features produced by a fully trained model should be very similar among different domains in terms of their distribution. By utilizing the recently proposed backpropagation architecture to implement adversarial domain adaptation [Ganin and Lempitsky, 2014], both the model and the adversary can be trained jointly. We note that previous implementations of this idea involved a multistep process training process; one step for the classifier and one step for the adversary.

So far, adversarial domain adaptation has seen success on popular computer vision datasets such as MNIST [Ganin and Lempitsky, 2014] and the idea has been used with limited success on the natural language processing tasks such as text classification [Liu et al., 2017] and speech recognition [Sun et al., 2017]. As part of this thesis, we

will explore the novel idea of applying adversarial domain adaptation to the stance detection problem. To the best of our knowledge, this technique has never been applied to the stance detection task or any of the datasets used in this thesis.

# Chapter 3

# The Fact Checking Problem

In this chapter, we will discuss our work on the fact checking problem. As discussed earlier in Section 1.2, this problem involves classifying an input claim $c$ based on the veracity of that claim. To compare the effectiveness of our ideas on this task to other techniques, we evaluate models on the LIAR dataset [Wang, 2017]. We will begin by examining the dataset before explaining our experiments and results.

## 3.1 Dataset Description

The LIAR dataset contains claims, their labels, and some metadata for each claim including information about the speaker, subject, party affiliation, and credit history. At the time of these experiments, this was a new and relatively unexplored dataset whose size is more than an order magnitude larger than any other similar dataset available at the time. The dataset is very balanced in terms of label classes and the data is drawn from PolitiFact[1], generally known as a reputable source.

First, we will examine some basic properties of the LIAR dataset. The average number of words for each of these claims is 18.0. Each of the claims are labeled by one of the six following labels in order of least true to most true: *pants-fire*, *false*, *barely-true*, *half-true*, *mostly-true*, and *true*. Examples of claims corresponding to each label can be found in Table 3.1. The distribution of labels is fairly balanced as

[1] http://www.politifact.com/

| Label | Claim |
|---|---|
| pants-fire | Says that in a hearing, Rep. Gabrielle Giffords suggested to Gen. David Petraeus that the Army put more emphasis on less environmentally damaging methods, like stabbing or clubbing enemy forces in order to minimize the carbon output. |
| false | Under the Cash for Clunkers program, all we've got to do is ... go to a local junkyard, all you've got to do is tow it to your house. And you're going to get $4,500. |
| barely-true | Says Rush Limbaugh made it clear hed rather see the country fail than President Barack Obama succeed. |
| half-true | The Social Security trust fund is sound. Without anything being done, it would function well into 2038; and even after that time with no changes, we could pay 80 percent of the benefits that people have earned. |
| mostly-true | High school students arrested on campus are twice as likely not to graduate and four times less likely to graduate if they've appeared in court. |
| true | Says George LeMieux was one of two Republicans who voted for President Barack Obama's jobs bill. |

Table 3.1: Examples of claim, label pairs from the LIAR dataset.

shown in Table 3.2, with the exception of the *pants-fire* label which has significantly fewer occurrences. Between the train, valid, and test partitions, the distributions are also fairly similar.

According to the metadata available in the dataset, there are 144 distinct subjects covered by the claims in the LIAR dataset, with the top five most common ones being 'economy', 'health-care', 'taxes', 'federal-budget', and 'education.' Claims may have multiple different subjects associated with them if appropriate. Each subject occurs an average of 193 times overall in the dataset.

There are $3,312$ distinct speakers in the LIAR dataset with the top five most common ones being 'barack-obama', 'donald-trump', 'hillary-clinton', 'mitt-romney', and 'john-mccain'. Each distinct speaker has an average of 3.9 claims in the dataset. The claims also have a fairly even distribution of party affiliation between Democrat and Republican parties throughout the train, validation, and test data set as shown in Figure 3.3.

| Label | Overall | Train | Valid | Test |
|---|---|---|---|---|
| pants-fire | 1049 | 841 | 116 | 92 |
| false | 2508 | 1996 | 263 | 249 |
| barely-true | 2106 | 1657 | 237 | 212 |
| half-true | 2632 | 2119 | 248 | 265 |
| mostly-true | 2458 | 1966 | 251 | 241 |
| true | 2059 | 1682 | 69 | 208 |

Table 3.2: Label frequency distribution for the LIAR dataset.

There is also credit history metadata for each claim. This metadata contains total counts of *pants-fire*, *false*, *barely-true*, *half-true*, and *mostly-true* for each speaker. It is important to note that the *true* label counts are missing; the reason for this is unclear. As mentioned in the original LIAR paper [Wang, 2017], the label corresponding to the claim being examined must be removed from these counts before using the credit metadata. Otherwise, because many speakers have only one or a few total claims, the credit metadata would act as almost a direct mapping to the label of the claim.

| Party | Overall | Train | Valid | Test |
|---|---|---|---|---|
| Republican | 5675 | 4507 | 597 | 517 |
| Democrat | 4144 | 3343 | 395 | 406 |
| None | 2183 | 1746 | 223 | 214 |

Table 3.3: Party affiliation frequency distribution for the LIAR dataset.

To illustrate, if the current label is not removed from the credit metadata, then a naive maximum likelihood model using the credit metadata will result in accuracies of $43.03\%, 45.48\%, 43.09\%$ for the training data, validation data, and test data respectively. This is a significantly better performance than any baseline model can perform. If the label of the current statement is removed from the credit metadata, the same maximum likelihood model yields $21.87\%, 24.61\%, 23.67\%$ accuracy for the training, validation, and test datasets. This is much more reasonable. After this modification, the credit metadata can be used without being unrealistically predictive of the label.

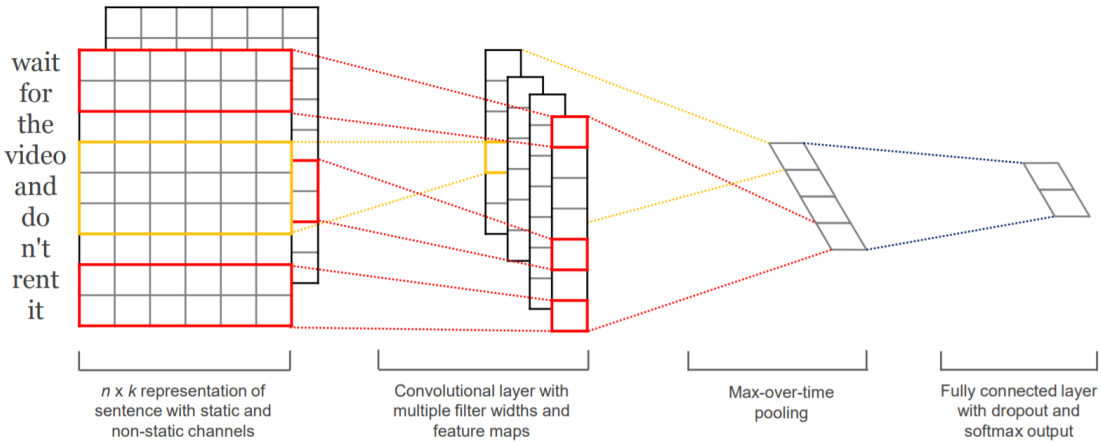


Figure 3-1: Model of a generic CNN as presented in [Kim, 2014]. The input sentence is first converted into a matrix of word vectors. Then, kernels of filter size 2 and 3 are applied, with multiple filters per filter size. The vectors are then pooled and the results are fully connected to a hidden layer.

## 3.2 Method

For our experiments, we used the Convolutional Neural Network (CNN) architecture, Bidirectional Long Short Term Memory (Bi-LSTM) architecture, and a stacked bidirectional LSTM and CNN architecture (Bi-LSTM+CNN). These are similar to the models that were implemented in the original LIAR paper [Wang, 2017], which we use as baselines for comparison.

### 3.2.1 CNN Architecture

Originally created to be used for image processing problems, CNNs have found effective application in various NLP applications when applied to word embedding matrices [Kim, 2014]. Generally speaking, CNNs contain at least one convolutional layer which takes the input and convolves it over a kernel.

If we define $n$ to be the size of the word vector, the convolution kernels are generally of dimensions $i$ by $n$, where $i$ is a positive integer, which produce outputs representing $i$-gram models. For example, if the size of the word vector is 300, and the kernel size is 2 by 300, the convolution will convolve two word vectors at a time and output

information corresponding to every consecutive pair of words. In other words, it captures bigram information. Often, more than one kernel size will be used to capture dependencies amongst different numbers of consecutive words.

After convolving the input, a simple but powerful architecture involves connecting the output of the convolutional layer to a hidden layer which is then connected to an output layer. A diagram of such a model, borrowed from previous work [Kim, 2014], is shown in Figure 3-1.

### 3.2.2 Bi-LSTM Architecture

An older, but equally effective approach to classifying textual content is to use a Long Short-Term Memory (LSTM) model [Hochreiter and Schmidhuber, 1997]. A LSTM is similar to the Recurrent Neural Network (RNN) in that it leverages information from previous inputs for processing the current input. In general for natural language applications, individual words are passed in one at a time to RNNs and the recurrent structure allows each word to utilize information about the previous words. However, RNNs struggle to handle longer sequences of inputs as the information from previous inputs are lost quickly. LSTMs counteract this information loss by using memory units, which can selectively save input information within the cell itself. An example diagram of a LSTM cell can be found in Figure 3-2.

Bidirectional LSTMs are an extension of LSTMs and involve using two LSTMs at once; one which takes sentence input in forward order and the other in backwards order [Graves and Schmidhuber, 2005]. This way, for each input, the model gets sequential information for inputs before and after it. With this extra information, bidirectional LSTMs are able to outperform LSTMs for many tasks [Graves and Schmidhuber, 2005].

### 3.2.3 Stacked LSTM and CNN Model Architecture

Although the bidirectional LSTM model is capable of remembering information over a long sequence of inputs, it's not able to capitalize on the n-gram coherence of the

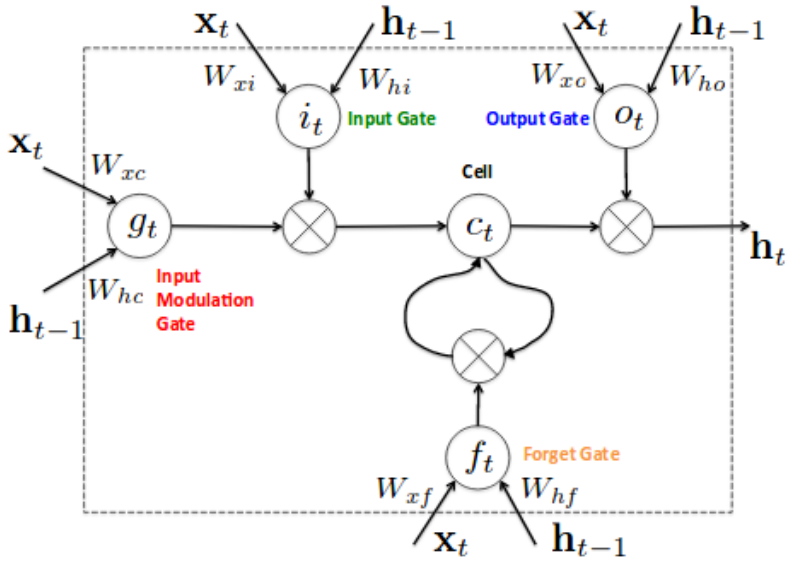


Figure 3-2: Model of a generic LSTM cell as presented by Chen [Chen, 2016]. LSTM cells contain a input and output gates of a LSTM cell control information flow to and from the memory unit. Additionally, LSTM cells have a forget gate, which can be used to remove information from the memory cell.

CNN model. In order to take advantage of the strengths of both models, a stacked bidirectional LSTM and CNN model can be used. The model is created by modifying the previous bidirectional LSTM model to process the input through a bidirectional LSTM layer followed by a CNN layer before being fully connected to a hidden layer.

It's been found that for some applications, using this combination of layers produces better results than using just one. For example, for the problem of question answer matching [Tan et al., 2016], the stacked LSTM and CNN model outperforms both the LSTM only model and the CNN only model. For our work, we will expand on this model by using a bidirectional LSTM rather than just a LSTM as we believe that the extra sequential information will benefit the model. An example of a stacked LSTM and CNN architecture is shown in Figure 3-3.

### 3.2.4 Evaluation Metrics

We use the following evaluation metrics to evaluate our models:

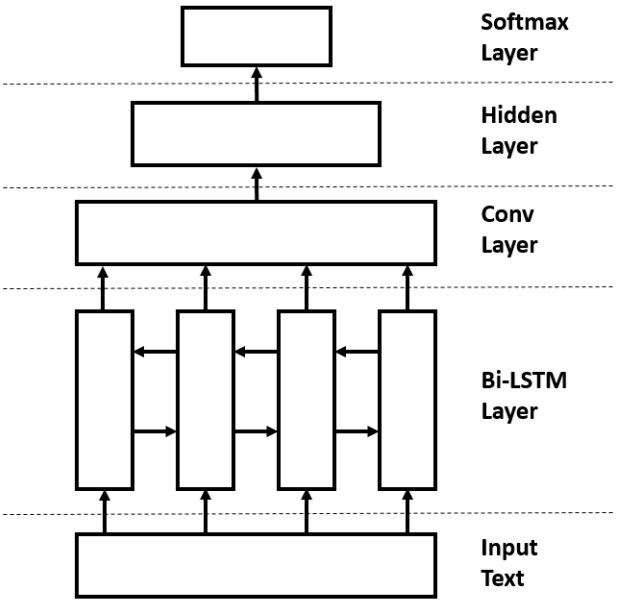


Figure 3-3: Diagram of Bi-LSTM and CNN model. The input layer is first fed into the Bi-LSTM layer. The features extracted from the Bi-LSTM layer are then fed into the convolutional layer. Finally, the output from the convolutional layer is fed into a fully connected hidden layer before undergoing classification via a final softmax layer.

(i) **Average Accuracy**: The accuracy of models averaged over multiple trainings. Accuracy is defined as the number of correctly classified examples divided by their total number of examples.

(ii) **Best Accuracy**: The accuracy of the trained model with highest validation accuracy. This metric is to allow direct comparison to the results of the original LIAR paper.

## 3.3 Experiments

For the LIAR task, we optimized baseline models using CNN and bidirectional LSTM architectures in combination with the given metadata. We also experimented with the novel idea of applying the stacked bidirectional LSTM (Bi-LSTM) and CNN model to this task. After tuning some hyperparameters, each of these models exceeds the original LIAR paper significantly in terms of accuracy. Each of the models described

next in this section were implemented using the Keras deep learning library.

### 3.3.1 CNN Experiments

For our models involving CNNs, we imitated and improved upon the basic construction described in the original LIAR paper [Wang, 2017] with several improvements to improve performance. That construction was inspired by a recently proposed model for using CNNs on text classification problems [Kim, 2014] as discussed in Section 3.2.1.

First, our model converted the input claim into a matrix of word vectors. Although the original paper used Word2Vec [Mikolov et al., 2013a] vectors trained on Google News[2], we discovered that GloVe [Pennington et al., 2014] vectors performed better for the task, particularly the pretrained 840B token common crawl set.[3] That matrix of embeddings was then subject to a Keras embedding layer with embedding training enabled. This trained the embeddings to be more specialized for the LIAR dataset, slightly improving the overall performance.

Then, the model convolves the input matrix on filter sizes of 2, 3, and 4 with 128 filters for each filter size. This effectively extracts bigram, trigram and quadgram information about the input claim. Alternative filter amounts and sizes were tried, but no arrangement found improved upon the current setting. A ReLU activation is used for the convolution. After, the convolution output is subject to average pooling. The original construction employed max pooling but we found that average pooling gives much better results. After these operations, a single value is output from each filter.

The model then concatenates all of the output values together into a single vector. That vector is then subject to dropout at a rate of 0.2. If any metadata is used, it is then concatenated to the resulting vector. Finally, the model connects the vector to a fully connected hidden layer with 100 neurons. That layer is then fed into a softmax layer which predicts the claim label. Both the dropout rate and hidden layer

[2]https://github.com/mmihaltz/word2vec-GoogleNews-vectors
[3]https://nlp.stanford.edu/projects/glove/

size chosen as a result of tuning.

We trained each model with 20 epochs using the `Adam` optimizer rather than the traditional Stochastic Gradient Descent which the paper originally used. Using `Adam` allowed us to use much fewer training epochs and actually allowed training to converge onto better models. The model with lowest validation loss was chosen from training. The model almost always started overfitting to the training data by epoch 5, so fewer training epochs could have been used. The accuracy results for our models and the original reported results can be found in Figure 3.4. Using a TITAN X GPU, each model took less than 5 minutes to train.

### 3.3.2 Bidirectional LSTM Experiments

For the bidirectional LSTM model (Bi-LSTM), we also converted the input claim into GloVe word vectors and passed the matrix into a Keras embedding layer. The output was then subject to a bidirectional LSTM layer. After tuning, we decided that input and recurrent dropout rates of 0.4 and an output dimension of 64 for the LSTM were approximately optimal. Note that because the LSTM was bidirectional, the actual output dimension was $64 * 2 = 128$.

The output vectors after every word was processed through the LSTM were saved and concatenated into one large vector, which was then subject to a dropout at a rate of 0.2. Similar to the CNN model, any metadata was concatenated at this step and the result was fully connected to a 100 neuron hidden layer. That hidden layer is then connected to a softmax layer which predicts the label of the input claim. As before, this model was trained with 20 epochs using the Adam optimizer. Using a TITAN X GPU, each model took less than 15 minutes to train.

### 3.3.3 Stacked Bidirectional LSTM and CNN Experiments

The Stacked Bidirectional LSTM and CNN model (Bi-LSTM+CNN) takes the matrix of word embeddings from the Keras embedding layer and first processes it through a Bidirectional LSTM layer. After tuning, the same set of parameters for the Bidirec-

tional LSTM also had good results for the combined model; the dropout rates were set to 0.4 and the output dimension was set to 64 for one direction of the LSTM.

Then, the output of the LSTM was passed into the convolutional layer. Again, it turns out that the same parameters for the convolutional layer as the CNN model produced strong results. To reiterate, the filter sizes chosen were 2, 3, and 4 and each filter size had 128 filters. The dropout rate was 0.2 in the convolutional step and average pooling was used. The output of the convolutional layer along with any metadata is then passed through a dropout layer of rate 0.2 and then fully connected to a hidden layer of size 100. Finally, the hidden layer is connected to the softmax layer which predicts the labels. The model was trained with 20 epochs with the Adam optimizer. Using a TITAN X GPU, each model took less than 15 minutes to train.

## 3.4 Analysis

The results of our experiments are shown in Table 3.4. Our CNN model with any subset of metadata outperformed the original reported results by 1% to 5% across the board in terms of validation accuracy. This lead to a higher test accuracies in general. Furthermore, these results are in line with those reported before, indicating that the model produces consistent results. Next, for the Bi-LSTM model, our models outperformed the previously reported accuracies by more than 5% each. Our results show that the Bi-LSTM model is actually comparable to the CNN models, despite the results that the original paper indicating otherwise. Finally, our stacked Bi-LSTM and CNN model generally performed similarly to the CNN model. This makes sense since the CNN layer was the last layer before the multilayer perceptron used for label prediction. It indicates that the LSTM layer prior to that CNN layer did not provide any additional useful information.

| Model | | Best Val | Best Test | Avg Val | Avg Test |
|---|---|---|---|---|---|
| 1. | Wang-CNN | 26.0 | 27.0 | N/A | N/A |
| 2. | Wang-CNN+subject | **26.3** | 23.5 | N/A | N/A |
| 3. | Wang-CNN+party | 25.9 | 24.8 | N/A | N/A |
| 4. | Wang-CNN+credit | 24.6 | 24.1 | N/A | N/A |
| 5. | Wang-CNN+all | 24.7 | **27.4** | N/A | N/A |
| 6. | Wang-Bi-LSTM | 22.3 | 23.3 | N/A | N/A |
| 7. | CNN | 28.0 | 27.4 | 27.0 | 26.3 |
| 8. | CNN+subject | 27.8 | 26.9 | 26.3 | 26.8 |
| 9. | CNN+party | 30.4 | 27.9 | 28.3 | 27.0 |
| 10. | CNN+credit | 29.0 | 27.5 | 27.4 | 27.8 |
| 11. | CNN+all | **30.0** | **28.6** | **28.8** | **28.2** |
| 12. | Bi-LSTM | 27.5 | 24.6 | 26.4 | 25.8 |
| 13. | Bi-LSTM+subject | 27.3 | 26.8 | 26.1 | 25.8 |
| 14. | Bi-LSTM+party | **28.8** | **26.8** | 26.6 | 26.2 |
| 15. | Bi-LSTM+credit | 27.2 | 25.8 | 26.0 | 26.7 |
| 16. | Bi-LSTM+all | 28.3 | 26.0 | **26.8** | **26.3** |
| 17. | Bi-LSTM+CNN | 28.0 | 28.0 | 27.0 | 26.8 |
| 18. | Bi-LSTM+CNN+subject | 27.7 | 27.0 | 26.8 | 27.2 |
| 19. | Bi-LSTM+CNN+party | 29.9 | 26.3 | 28.5 | 26.6 |
| 20. | Bi-LSTM+CNN+credit | 28.8 | **28.4** | 27.7 | 27.6 |
| 21. | Bi-LSTM+CNN+all | **31.0** | 27.6 | **28.9** | **28.1** |

Table 3.4: Model Accuracy Results. The 'Wang' prefix denotes results from original LIAR paper [Wang, 2017]. Suffixes subject, party, credit, and all indicate that the corresponding pieces of metadata were used. Best model was picked by choosing the highest validation accuracy model out of 10 trained. Avg indicates average accuracy over 10 trained models.

## 3.5 Discussion

In addition to the provided metadata, we also experimented with using other metadata such as part of speech frequency. However, we did not see any significant improvements with this metadata; in fact performance decreased when it was used. We also tried other model constructions, such as a parallel rather than a stacked Bi-LSTM and CNN model and a stacked CNN Bi-LSTM model in which the CNN layer came first. Both of these models did not improve the performance.

Though it might be possible that the methods mentioned can be used in a more

effective way for this task, we believe that even a fully optimized model using these techniques will not significantly exceed the results we presented here. Part of the problem is that the problem space is simply too large for the amount of data provided by the LIAR dataset. Possible approaches to resolve this issue include using additional data or innovating new models which can better utilize existing information. Another possible approach is to break down the fact checking problem into multiple smaller tasks, each with a smaller problem space following a breakdown mentioned in Section 1.2. Then, existing data might be better leveraged to solve these tasks with smaller problem spaces before being combined to create a fact checking system.

In addition, a particular issue with this data is about lack of clarity on how the data labels are defined; the line between adjacent label classes such as barely-true and half-true are especially unclear even for humans. So, it will also be hard for a machine trained on LIAR data to make this distinction as well. Likely, a disambiguation of the label classes will allow for better performance.

Because we could not overcome these challenges for the LIAR task, we pivoted to focus our work on a smaller subproblem of fact checking known as stance detection. This problem has the advantage of having a much small problem space as well as a larger set of available data. We will discuss our experiments regarding stance detection in the next chapter.

## 3.6 Summary

We presented CNN and bidirectional LSTM (Bi-LSTM) based models for the LIAR dataset which significantly exceeded previous baselines. These results show that both neural models have similar potential in the task, contrary to what previous results showed. Also, we presented other model ideas we tried, such as the stacked CNN and Bi-LSTM model, though these more complex models did not see any significant performance gain.

# Chapter 4

# Stance Detection using Adversarial Domain Adaptation

In this chapter, we discuss our work on the stance detection problem. As discussed earlier in Section 1.2, this problem involves automatically determining the perspective of a document relative to a claim. Here, we focus our experiments on the novel idea of applying adversarial domain adaptation to address this problem, particularly when the labeled data is limited.[1] The upcoming sections will discuss our datasets (Section 4.1), method (Section 4.2), experiments (Section 4.3) and findings (Section 4.4).

## 4.1 Datasets

For the stance detection problem, we primarily focus on using the Fake News Challenge (FNC) dataset [Pomerleau and Rao, 2017], a publicly available stance detection dataset, to evaluate our models. Since the dataset has very limited labeled data for some label classes, we supplement it with data from the Fact Extraction and VERification (FEVER) [Thorne et al., 2018] dataset, an independent stance detection dataset (see Section 4.1.2), and with data from the Stanford Natural Language Inference (SNLI) [Bowman et al., 2015] dataset, which was created for a related task

[1] Results from a portion of these experiments were published in the 2018 NIPS Continual Learning Workshop [Xu et al., 2018].

| Label | Claim | Document |
|---|---|---|
| agree | Hundreds of Palestinians flee floods in Gaza as Israel opens dams | Hundreds of Palestinians were evacuated from their homes Sunday morning after Israeli authorities opened a number of dams near the border ... |
| disagree | Spider burrowed through tourist's stomach and up into his chest | Fear not arachnophobes, the story of Bunbury's "spiderman" might not be all it seemed. Perth scientists have cast doubt over claims that a spider burrowed into a man's body during his first trip to Bali ... |
| discuss | Soon Marijuana May Lead to Ticket, Not Arrest, in New York | Law-enforcement officials tell the New York Times that soon the NYPD may issue tickets for low-level marijuana possession rather than making arrests ... |
| unrelated | N. Korea's Kim has leg injury but in control | You want a gold Apple Watch, you say? Then it's going to cost you... a lot. The vanilla variant of Apple's newest wrist-worn wearable device only costs $349 ... |

Table 4.1: Randomly chosen examples of claim-document pairs and their labels from the FNC dataset. The relevant snippet from each document is displayed.

(i.e., textual inference problem; see Section 4.1.3). Next, we will discuss each of these datasets in depth.

### 4.1.1 Fake News Challenge

The FNC dataset contains claim-document pairs, with labels which explain the stance relationship between them. The possible labels are *agree*, *disagree*, *discuss*, and *unrelated*. Randomly selected samples from the FNC dataset are displayed in Table 4.1. The dataset is partitioned into train and test sets. Combined, there are about 75k claims in total, and the train set has approximately twice as many claims as the test set. Each claim is on average about 11 words long, and each corresponding document is about 360 words long. As shown in Table 4.2, there is a significant imbalance of the frequency of each label class. In particular, the *agree* and *disagree* classes have

| Label | Overall | Train | Test |
|---|---|---|---|
| agree | 5,581 | 3,678 | 1,903 |
| disagree | 1,537 | 840 | 697 |
| discuss | 13,373 | 8,909 | 4,464 |
| unrelated | 54,894 | 36,545 | 18,349 |
| total | 75,485 | 49,972 | 25,413 |

Table 4.2: Label frequency distribution for the FNC dataset.

relatively few data points. Comparing the train and test sets, the label distributions are approximately the same. However, there is a slightly higher proportion of *disagree* labels in the test set.

### 4.1.2 Fact Extraction and VERification

The Fact Extraction and VERification (FEVER) dataset is constructed such that a claim is associated with some number of related Wikipedia documents, each document is assigned as either *SUPPORTS* or *REFUTES* label based on its stance relationship to the claim. The claim is then assigned an overall label, either *SUPPORTS* or *REFUTES* based on its relationship to all of the associated evidence documents. If there are no associated documents, the claim is labeled as *NOT ENOUGH INFO*. Some randomly chosen examples from the FEVER dataset are displayed in Table 4.3. As shown in Table 4.4, there are about 80k *SUPPORTS* claims, 30k *REFUTES* claims, and 35k *NOT ENOUGH INFO* claims. Since there are no associated documents with *NOT ENOUGH INFO* claims, we end up not using these labels.

In our work, we use the FEVER dataset to supplement the FNC dataset using the domain adaptation technique by hypothesizing that FEVER *SUPPORTS* and *REFUTES* labels are similar to FNC *agree* and *disagree* labels respectively. However, we have to reformat the FEVER dataset slightly to make it similar to the format of the FNC dataset in which each example consists of a single document and a single claim. Thus, we break up any claim that is associated with more than a single document into multiple claim-document pairs. For example, the claim shown in the

| Label | Claim | Document |
|---|---|---|
| SUPPORTS | Felicity Jones was in a 2014 British biographical romantic drama movie. | 1. *"Felicity_ Jones"*: In 2014, her performance as Jane Hawking in The Theory of Everything also met with critical acclaim, garnering her nominations for ...<br>2. *"The_ Theory_ of_ Everything_ (2014 _film)"*: The Theory of Everything is a 2014 biographical romantic drama film which is set at Cambridge University ... |
| REFUTES | The People vs. Larry Flynt is an unproduced screenplay. | 1. *"The_ People_ vs._ Larry_ Flynt"*: The People vs. Larry Flynt is a 1996 American biographical drama film ... |

Table 4.3: Randomly chosen examples of data from the FEVER dataset. The name of each document associated with the claim is given in italics, followed by the relevant snippet from the document.

| Label | Train | Dev | Test |
|---|---|---|---|
| SUPPORTS | 80,035 | 3,333 | 3,333 |
| REFUTES | 29,775 | 3,333 | 3,333 |
| NOT ENOUGH INFO | 35,639 | 3,333 | 3,333 |
| total | 145,449 | 9,999 | 9,999 |

Table 4.4: Label frequency distribution for the FEVER dataset.

first row of Table 4.3 is associated with two documents, and is converted into two separate claim-document pairs. After this modification, we end up with more data points than officially reported by FEVER [Thorne et al., 2018] with around 102k *SUPPORTS* labels and around 37k *REFUTES* labels from the train dataset.[2] The FEVER average claim and document lengths are respectively 8 and 313 words which is almost the same as FNC as explained in Section 4.1.1.

Note that, the FEVER dataset also contains some annotations at the sentence level for each document with respect to its corresponding claim. However, we use the

[2]As the FEVER development and test datasets had not been released at the time of these experiments, we have not used those datasets.

| Label | Claim | Document |
|---|---|---|
| entailment | A person on a horse jumps over a broken down airplane. | A person is outdoors, on a horse. |
| neutral | A person on a horse jumps over a broken down airplane. | A person is training his horse for a competition. |
| contradiction | A person on a horse jumps over a broken down airplane. | A person is at a diner, ordering an omelette. |

Table 4.5: Randomly chosen examples of data from the SNLI dataset. The premise-hypothesis pairs are considered as the claim-document pairs.

| Label | Train | Dev | Test |
|---|---|---|---|
| entailment | 183,416 | 3,329 | 3,368 |
| contradiction | 183,187 | 3,278 | 3,237 |
| neutral | 182,764 | 3,237 | 3,219 |
| total | 549,367 | 9,844 | 9,824 |

Table 4.6: Label frequency distribution for the SNLI dataset.

entire document when we supplement the FNC dataset as the documents in FNC are long length. In contrast, we use the FEVER annotations at the sentence level with the SNLI dataset as the SNLI data consists of sentences (see Section 4.4).

### 4.1.3 Stanford Natural Language Inference

The Stanford Natural Language Inference (SNLI) dataset was created for the textual inference problem and is significantly larger than either the FNC or FEVER datasets, with about 550k total data points. This dataset contains sentence-sentence pairs (premise-hypothesis pairs) with labels to show whether the two input sentences warrant an *entailment*, *contradict*, or *neutral* relationship to each other. Some randomly chosen examples of the SNLI dataset are displayed in Table 4.5. Also, as shown in Table 4.6, the dataset is well balanced with around 183k of data points for each label.

The task of this dataset (i.e., textual inference) is different from the task of the

FNC and FEVER datasets (i.e., stance detection). However, the *entailment* and *contradiction* labels in SNLI can be considered as being similar to the *agree* and *disagree* labels in the FNC dataset and the *SUPPORTS* and *REFUTES* labels in the FEVER dataset. It's also possible to interpret *neutral* label in SNLI to be similar to *discuss* in FNC, although the connection is not as strong. In this chapter, we also investigate the impact of using the SNLI dataset through a domain adaptation technique to supplement the FNC and FEVER datasets.

We consider the premise-hypothesis pairs in SNLI as the claim-document pairs. This means both the claim and the document in SNLI are only a single sentence. This results in a notable difference between the average length of the documents in SNLI, around 14 words, and the documents in the FNC and FEVER datasets, a couple hundred words, making it harder to leverage for domain adaptation. Ideally, though, our models can pick some features that are independent of the length of the documents, so that the SNLI data may be able to help improve these features.

## 4.2 Method

Previously proposed approaches for stance detection generally contain two components [Baird et al., 2017, Hanselowski et al., 2017, Riedel et al., 2017]: a feature extraction component followed by a class label prediction component. In this thesis, we present a model for stance detection that augments the traditional models with a third component: a domain adaptation component.

Our domain adaptation component uses adversarial learning [Ganin and Lempitsky, 2014] to encourage the feature extraction component to select common, rather than domain specific, features when input data is from multiple different domains. This allows the model to better leverage *source* domain for better prediction on data from *target* domain.

The general architecture of our model is shown in Figure 4-1. As illustrated, the inputs are first given to the "*Feature Extraction Component*" to compute their features and representations. These features are then passed to the "*Label Prediction*

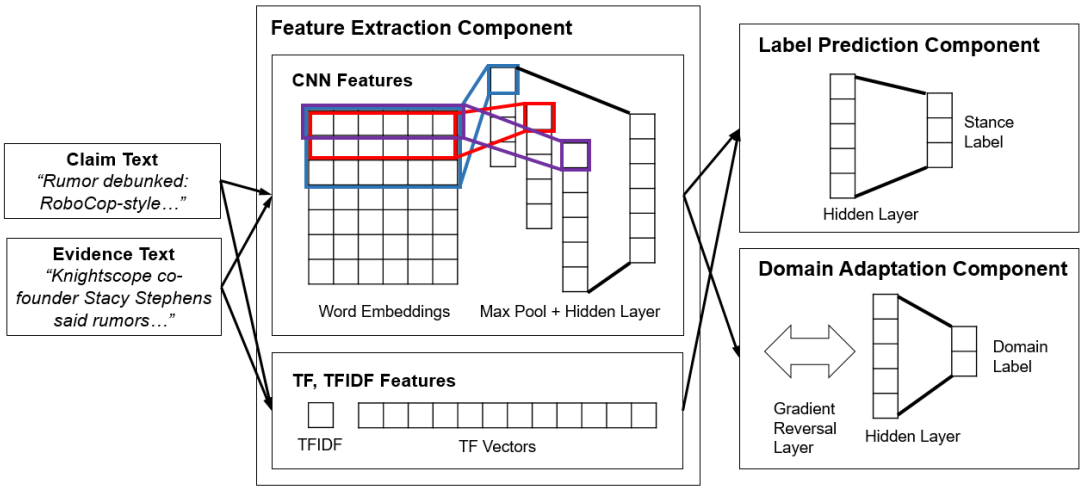


Figure 4-1: The architecture of our model for stance detection which uses adversarial domain adaptation.

*Component*" and then to the "*Domain Adaptation Component*." In the model, while both latter components try to minimize their own losses, the feature extraction component attempts to maximize domain classification loss to encourage better mixture of examples from different domains. The components of the model are described in detail below.

### 4.2.1 Feature Extraction Component

This component takes the input claim $c$ and document $d$ and converts them to their semantic representations and features. To do this, we use bag of words (BOW) related features, essentially TF and TF-IDF weighted features. We select these BOW features as they are useful to filter documents with *unrelated* stance labels as we will show in Section 4.4. Furthermore, we also use a convolutional neural network (CNN) approach discussed in Section 3.2.1 for learning representations of claims and documents. To recap, we use a CNN because it can capture $n$-grams and long range dependencies [Yu et al., 2014], and can extract discriminative word sequences that are common in the training instances [Severyn and Moschitti, 2015]. These traits make CNNs useful for dealing with long documents [Mohtarami et al., 2016].

### 4.2.2 Label Prediction Component

The label prediction component uses a multilayer perceptron (MLP) with a fully connected hidden layer followed by a *softmax* layer which employs *cross entropy* loss as the cost function. This component will predict stance labels as *agree*, *disagree*, *discuss*, or *unrelated* for a given set of claim and document features.

### 4.2.3 Domain Adaptation Component

The domain adaptation component contains a domain classifier which includes a MLP followed by a *softmax* layer. Given a set of features for a claim-document pair, the domain classifier predicts which domain the features originated from.

The domain classifier acts as an adversary because the model is constructed to encourage the feature extraction component to maximize the domain classifier loss, while the domain adaptation component itself attempts to minimize that same loss. This is because a high domain classifier loss implies that the domain classifier is unable to accurately discern whether a set of features belongs to the *source* or the *target* domain. This implies that the features extracted from the input examples are common to both the *source* and *target* domains, which is the ultimate goal of the adversarial domain adaptation technique.

To achieve the desired adversarial behavior, the features from the feature extraction component are passed to a *gradient reversal layer* before being passed to the domain classifier. The gradient reversal layer is a simple identity transform during forward propagation and multiplies the gradient by a negative constant (the gradient reversal constant $\lambda$) during backpropagation [Ganin and Lempitsky, 2014]. In particular, if we define the downstream domain loss to be $L_d$ and the upstream parameters to be $\theta_f$, the gradient reversal layer essentially replaces the partial derivative $\frac{\partial L_d}{\partial \theta_f}$ with $-\lambda \frac{\partial L_d}{\partial \theta_f}$. Effectively, this encourages the upstream components to maximize $L_d$ while the downstream domain classifier tries to minimize $L_d$. The desired adversarial training behavior can be achieved through normal model training.

In order to avoid noise from the domain classifier during initial stages of training, $\lambda$

is generally set to a schedule which starts small and increases steadily during training with:

$$\lambda_p = \frac{2}{1 + exp(-\gamma \times p)} - 1 \tag{4.1}$$

where $p$ represents the fraction of training epochs elapsed. This allows the domain classifier to train properly initially before being influenced by the gradient reversal layer. Meanwhile, the learning rate starts high and decays over time with:

$$\mu_p = \frac{\mu_0}{(1 + \alpha \times p)^\beta} \tag{4.2}$$

where $p$ represents the fraction of training epochs elapsed. The learning rate schedule has the dual purpose of encouraging conversion as well as allowing for strong initial domain classifier training. The specific settings for the parameters $\gamma$ in domain adaptation constant, and $\alpha$ and $\mu_0$ in learning rate formulas are described in Section 4.3.

### 4.2.4 Model Parameters and Training Procedure

For our CNN model, we use 300-dimensional word embeddings from Word2Vec [Mikolov et al., 2013b], which were trained on Google News, and 100 feature maps with filter width {2, 3, 4}. We set the maximum word lengths of the input claims and documents to be 50 and 500 respectively, where these values are close to the average length for claims and documents in the target train data. For the BOW model, we keep the hyperparameters and features to be the same as the baseline model [Riedel et al., 2017].

Our models are trained using the `Adam` optimizer, and 20% of the train data is set aside as validation data. In the models with domain adaptation (DA) component, equal amounts of both *source* and *target* data are randomly selected at each epoch during training. Finally, we fine tune all the hyperparameters of our models on validation data which contains equal amounts of *source* and *target* data.

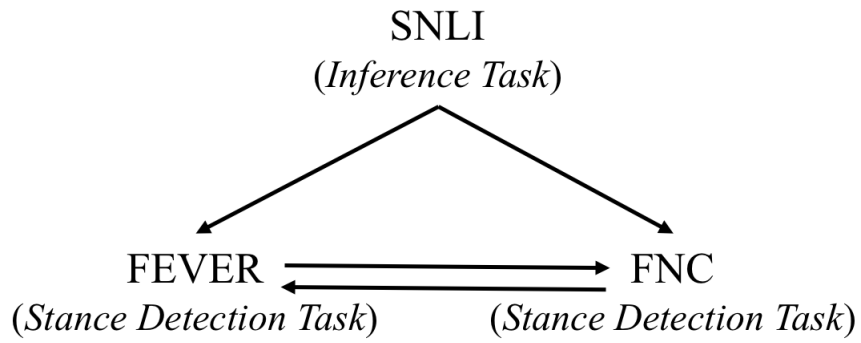


Figure 4-2: Exploring learning with adversarial domain adaptation across different domains (SNLI, FEVER, and FNC datasets) and tasks (stance detection and textual inference tasks).

To reduce domain classifier noise and promote convergence, the gradient reversal constant and learning rate are respectively defined as $\lambda_p$ and $\mu_p$ in Formula 4.1 and Formula 4.2 in Section 4.2.3. The parameters in the formulas are set to the values that used successfully by previous work [Ganin and Lempitsky, 2014] as $\gamma = 10$, $\mu_0 = 0.01$, $\alpha = 10$, $\beta = 0.75$.

## 4.3 Experiments

In this chapter, we aim to explore the effect of adversarial domain adaptation across different domains with both similar and dissimilar tasks. As Figure 4-2 shows, the similar tasks are based on stance detection where we consider FNC and FEVER datasets. The dissimilar tasks that we consider are stance detection (FNC or FEVER datasets) and textual inference (SNLI dataset) tasks. We first describe the baselines (Section 4.3.1), evaluation metrics (Section 4.3.2), our models with different configurations (Section 4.3.3), and then we present our results and analysis across these datasets (Section 4.4).

### 4.3.1 Baselines

When we use FNC as the target domain, we compare our domain adaptation (DA) model to the following previous work that applied their models on FNC data as target

domain:

(i) *Gradient Boosting*, which is the Fake News Challenge baseline, and it trains a gradient boosting classifier using handcrafted features reflecting polarity, refute, similarity and overlap between documents and claims.[3]

(ii) *TALOS* [Baird et al., 2017], which was ranked first at FNC. It uses a weighted-average between gradient-boosted decision trees (TALOS-Tree) and a deep convolutional neural network (TALOS-DNN).

(iii) *UCL* [Riedel et al., 2017], which was ranked third at FNC; this model trains a softmax layer using $n$-gram features (e.g., TF and TF-IDF). We compare with this model because our BOW model is similar to it and uses the same features.

Note that, since the FEVER data is a recently released dataset at the time of this work, there has not been any well known baselines available for the dataset. Thus, when we use FEVER data as the target domain, we only perform relative comparisons between our own models.

### 4.3.2 Evaluation Metrics

We use the following evaluation metrics to evaluate our models:

(i) **Macro-F1**: The average of the $F_1$ score for each class.

(ii) **Accuracy**: The number of correctly classified examples divided by their total number of examples.

(iii) **Weighted-Accuracy**: This metric is presented by Fake News Challenge[4] which is a two level scoring scheme. It gives 0.25 weight to the correctly predicted examples as *related* or *unrelated*. It further gives 0.75 weights to the correctly predicted *related* examples as *agree*, *disagree*, or *discuss*.

[3]The source code of the FNC baseline is available at `https://github.com/FakeNewsChallenge/fnc-1-baseline`

[4]Scorer is available at `www.fakenewschallenge.org`

### 4.3.3 Model Configurations

We present different variations of our models where each uses a subset of components and features shown in Figure 4-1. In particular, we try using different subsets of BOW features and CNN features during feature extraction and try both using and not using the domain adaptation component for each one. These variations help us to conduct ablation analysis on these information sources. The baseline and our models are trained to predict stance labels on target data; {*agree*, *disagree*, *discuss*, *unrelated*} for target FNC data and {*SUPPORTED*, *REFUTED*} for target FEVER data.

For the experiments with target FNC data, we further apply a two level *hierarchy* prediction scheme in our models, where the first level predicts whether the label is *related* or *unrelated*, and then the predicted *related* labels are passed to the second level to predict *agree*, *disagree*, and *discuss* labels. The reason of using this two level *hierarchy* scheme with target FNC data is that the current FNC models already perform well for the first level of the *hierarchy* scheme. For the first level, we use the BOW model which achieves an $F_1$ performance of 97.7% for *unrelated* and 93.9% for *related* labels. Then, the second level is trained with respect to the components and datasets used in the model.

We also conduct specific experiments for certain domain pairs (*source*→*target*): (*i*) For SNLI→FNC, we map the SNLI *neutral* labels to the FNC *discuss* labels to investigate if there is an impact. (*ii*) For SNLI→FEVER and FNC→FEVER, we balance the label distribution of the data over classes during training and (*iii*) we use the annotated sentences of a document that are specified as being relevant to the claim, rather than using the entire document.

| | Model | Source | Target | Weigh. Acc. | Acc. | Macro $F_1$ | $F_1$ Scores *agree/disagree/discuss/unrelated* |
|---|---|---|---|---|---|---|---|
| 1. | Gradient Boosting | - | FNC | 75.2 | 85.4 | 45.7 | 14.8 / 2.0 / 69.5 / 96.5 |
| 2. | TALOS *(#1st in FNC)* | - | FNC | 82.0 | 89.1 | 57.8 | 53.8 / 3.6 / 76.0 / 97.9 |
| 3. | TALOS-DNN | - | FNC | 60.8 | 66.5 | 41.8 | 27.6 / 9.3 / 47.4 / 82.7 |
| 4. | TALOS-Tree | - | FNC | 83.1 | 89.5 | 56.8 | 53.4 / 0.2 / 76.3 / 98.4 |
| 5. | UCL *(#3rd in FNC)* | - | FNC | 81.7 | 88.5 | 57.9 | 47.9 / 11.4 / 74.7 / 97.6 |
| 6. | BOW | - | FNC | 81.1 | 88.6 | 56.0 | 49.2 / 2.5 / 74.8 / 97.6 |
| 7. | CNN | - | FNC | 40.8 | 71.3 | 23.3 | 0.3 / 0.0 / 10.0 / 83.0 |
| 8. | BOW+CNN | - | FNC | 74.9 | 86.8 | 52.2 | 41.4 / 0.0 / 72.1 / 95.2 |
| 9. | BOW (hierarchy) | - | FNC | 80.7 | 88.5 | 57.3 | 49.5 / 7.5 / 74.3 / 97.7 |
| 10. | CNN (hierarchy) | - | FNC | 79.9 | 87.9 | 56.0 | 54.9 / 0.2 / 71.1 / 97.7 |
| 11. | BOW + CNN (hierarchy) | - | FNC | 80.3 | 88.2 | 56.5 | 56.0 / 0.0 / 72.1 / 97.7 |
| 12. | BOW | FEVER | FNC | 78.5 | 86.4 | 56.3 | 48.8 / 9.8 / 69.4 / 97.1 |
| 13. | [BOW + DA] | FEVER | FNC | 72.9 | 81.5 | 48.6 | 44.5 / 2.0 / 51.7 / 96.1 |
| 14. | BOW (hierarchy) | FEVER | FNC | 78.8 | 87.3 | 57.2 | 51.7 / 10.2 / 69.1 / 97.7 |
| 15. | [BOW + DA] (hierarchy) | FEVER | FNC | 78.5 | 87.1 | 56.4 | 51.6 / 8.3 / 68.0 / 97.7 |
| 16. | CNN | FEVER | FNC | 41.3 | 64.3 | 27.3 | 17.4 / 2.0 / 11.0 / 78.8 |
| 17. | [CNN + DA] | FEVER | FNC | 39.0 | 64.3 | 24.0 | 13.5 / 0.2 / 3.3 / 78.9 |
| 18. | CNN (hierarchy) | FEVER | FNC | 79.0 | 87.4 | 56.6 | 51.9 / 7.5 / 69.3 / 97.7 |
| 19. | [CNN + DA] (hierarchy) | FEVER | FNC | 79.1 | 87.7 | 57.9 | 51.2 / 11.4 / 71.3 / 97.7 |
| 20. | BOW + CNN | FEVER | FNC | 71.7 | 84.5 | 51.5 | 44.6 / 5.6 / 60.2 / 95.6 |
| 21. | BOW + [CNN + DA] | FEVER | FNC | 71.9 | 84.6 | 51.4 | 44.9 / 4.4 / 60.6 / 95.6 |
| 22. | BOW + CNN (hierarchy) | FEVER | FNC | 79.6 | 87.8 | 56.6 | 53.1 / 5.1 / 70.6 / 97.7 |
| 23. | BOW + [CNN + DA] (h.) | FEVER | FNC | **80.3** | **88.2** | **60.0** | **54.6 / 15.1 / 72.6 / 97.7** |

Table 4.7: FEVER → FNC. Results on the FNC test data for models trained using FNC training data and FEVER data. BOW, CNN and DA refer to our model when it uses bag of words features, convolutional features, and domain adaptation, respectively. When DA is present, square brackets indicate which features are passed to the domain adaptation component. The *hierarchy* in parentheses refers to our model with two level prediction scheme as explained in section 4.3.3. We show the results of the models based on the smallest loss for validation set across 5 independent runs.

## 4.4 Results and Analysis

### 4.4.1 Performance Analysis: FEVER → FNC

Table 4.7 shows the results of different models when FNC is considered as the target test data. Lines 1-5 show the results for baseline models explained in Section 4.3.1. As the results show, they perform weakly on *agree* and *disagree* stances, likely due to the small size of labeled data. Only 1.7% and 7.3% of the FNC data examples carry *disagree* and *agree* labels. Lines 6-23 show the results of different configurations of our model as explained in Section 4.3.3. First, we examine the models trained using only FNC data (lines 6-11). Lines 6-8 show the results for BOW, CNN and their combinations respectively; the results of these models with the *hierarchy* scheme are shown in lines 9-11. The results show that the *hierarchy* scheme helps models to perform better, especially for the CNN model where its result improves from 23.3% $F_1$ (line 7) to 56% (line 10). However, these models don't improve the baseline results, and they still struggle with identifying *agree* and *disagree* labels.

We then examine the impact of using the FEVER data on the performance of the model (lines 12-23) when FNC is considered as the target test data. Lines 12-15 show the results for the BOW model with and without the *hierarchy* scheme and domain adaptation (DA) component. The results show that domain adaptation using FEVER data does not improve the results for the BOW model. While the exact reason is unclear, perhaps this is because BOW features tend to be very discrete and extremely domain dependent. This makes it hard to transfer any useful BOW features across datasets.

We further repeat these experiments with the CNN model and, as shown in lines 16-19, the CNN model with the *hierarchy* scheme performs significantly better in terms of $F_1$ score when the domain adaptation component is used. Looking at the breakdown of $F_1$ scores for each label, this improvement is due to an improvement in the performance for *agree* and *disagree* labels. The difference in Macro $F_1$ between lines 18 and 19 shows that using domain adaptation can provide a significant benefit in this context.

Though the CNN results are improved, they are still worse than the baselines. To make more improvement, we propose to supplement the BOW features with the CNN features. In this case, the model can better leverage FNC data with the BOW features, while still retaining the boost from using FEVER data via the CNN features. Lines 20-23 show the results for our model when using a combination of the BOW and CNN models. As the results show, the combination model can boost performance for all metrics. The combination of the BOW and CNN models when using the FEVER dataset and domain adaptation technique has the best $F_1$ performance compared to all models including the baselines. This demonstrates that not only domain adaptation used in this manner can be effective, but it is good enough to exceed previous state of the art results. As expected, the *disagree* and *agree* classes have significant improvement when using FEVER dataset, since the FEVER data supplements only *agree* and *disagree* labels.

In summary, the results show:

- The *hierarchy* scheme can help our models to perform better across all metrics.

- Our best model is the combination of BOW, CNN+DA, and the hierarchy scheme. It outperforms the baselines, especially on the most important classes: *disagree* and *agree*.

- The *source* FEVER data can improve the performance of our model for target FNC data through adversarial domain adaptation, when uses CNN model (see lines 16-19 in Table 4.7) or BOW+CNN model (see lines 20-21 in Table 4.7).

### 4.4.2 Performance Analysis: SNLI → FNC

Table 4.8 shows the results of different models on the target FNC test data. Lines 1-5 show the results for baseline models and lines 6-23 show the results of different configurations of our model as explained in Section 4.3.3. The results of lines 1-11 are obtained when only the FNC train data is used, and lines 12-23 show the results

| | Model | Source | Target | Weigh. Acc. | Acc. | Macro $F_1$ | $F_1$ Scores *agree/disagree/discuss/unrelated* |
|---|---|---|---|---|---|---|---|
| 1. | Gradient Boosting | - | FNC | 75.2 | 85.4 | 45.7 | 14.8 / 2.0 / 69.5 / 96.5 |
| 2. | TALOS *(#1st in FNC)* | - | FNC | 82.0 | 89.1 | 57.8 | 53.8 / 3.6 / 76.0 / 97.9 |
| 3. | TALOS-DNN | - | FNC | 60.8 | 66.5 | 41.8 | 27.6 / 9.3 / 47.4 / 82.7 |
| 4. | TALOS-Tree | - | FNC | 83.1 | 89.5 | 56.8 | 53.4 / 0.2 / 76.3 / 98.4 |
| 5. | UCL *(#3rd in FNC)* | - | FNC | 81.7 | 88.5 | 57.9 | 47.9 / 11.4 / 74.7 / 97.6 |
| 6. | BOW | - | FNC | 81.1 | 88.6 | 56.0 | 49.2 / 2.5 / 74.8 / 97.6 |
| 7. | CNN | - | FNC | 40.8 | 71.3 | 23.3 | 0.3 / 0.0 / 10.0 / 83.0 |
| 8. | BOW+CNN | - | FNC | 74.9 | 86.8 | 52.2 | 41.4 / 0.0 / 72.1 / 95.2 |
| 9. | BOW (hierarchy) | - | FNC | 80.7 | 88.5 | 57.3 | 49.5 / 7.5 / 74.3 / 97.7 |
| 10. | CNN (hierarchy) | - | FNC | 79.9 | 87.9 | 56.0 | 54.9 / 0.2 / 71.1 / 97.7 |
| 11. | BOW + CNN (hierarchy) | - | FNC | 80.3 | 88.2 | 56.5 | 56.0 / 0.0 / 72.1 / 97.7 |
| | **SNLI Neutral = Discuss** | | | | | | |
| 12. | BOW (hierarchy) | SNLI | FNC | 79.7 | 87.8 | 56.3 | 44.8 / 10.0 / 72.5 / 97.7 |
| 13. | [BOW + DA] (hierarchy) | SNLI | FNC | 79.8 | 87.9 | 55.2 | 39.3 / 10.6 / 73.3 / 97.7 |
| 14. | CNN (hierarchy) | SNLI | FNC | 79.6 | 87.7 | 55.1 | 49.3 / 1.7 / 71.7 / 97.7 |
| 15. | [CNN + DA] (hierarchy) | SNLI | FNC | 79.3 | 87.6 | 52.5 | 33.7 / 4.5 / 74.0 / 97.7 |
| 16. | BOW + CNN (hierarchy) | SNLI | FNC | 80.6 | 88.4 | 57.7 | 48.7 / 10.2 / 74.0 / 97.7 |
| 17. | BOW + [CNN + DA] (h.) | SNLI | FNC | 80.5 | 88.4 | 56.0 | 42.6 / 9.3 / 74.2 / 97.7 |
| | **SNLI Neutral Unused** | | | | | | |
| 18. | BOW (hierarchy) | SNLI | FNC | 79.2 | 87.5 | 57.8 | 49.6 / 12.9 / 71.0 / 97.7 |
| 19. | [BOW + DA] (hierarchy) | SNLI | FNC | 78.6 | 87.2 | 58.7 | 49.6 / 17.9 / 69.4 / 97.7 |
| 20. | CNN (hierarchy) | SNLI | FNC | 79.1 | 87.5 | 56.5 | 49.4 / 8.0 / 70.8 / 97.7 |
| 21. | [CNN + DA] (hierarchy) | SNLI | FNC | 77.3 | 86.3 | 55.5 | 50.5 / 7.0 / 66.6 / 97.7 |
| 22. | BOW + CNN (hierarchy) | SNLI | FNC | 79.4 | 87.7 | 57.7 | 51.6 / 10.1 / 71.4 / 97.7 |
| 23. | BOW + [CNN + DA] (h.) | SNLI | FNC | 79.5 | 87.7 | 55.6 | 53.8 / 4.2 / 70.6 / 97.7 |

Table 4.8: SNLI → FNC. Results on the FNC test data for models trained using FNC training data and SNLI data. BOW, CNN and DA refer to our model when it uses bag of words features, convolutional features, and domain adaptation, respectively. When DA is present, square brackets indicate which features are passed to the domain adaptation component. The *hierarchy* in parentheses refers to our model with two level prediction scheme as explained in section 4.3.3. We show the results of the models based on the smallest loss for validation set across 5 independent runs.

of different configurations of our model when we use additional train data from a different domain and different task (i.e., SNLI) in addition to the FNC data. Lines 12-17 show the results when we map the *entailment*, *contradiction* and *neutral* labels in SNLI to the *agree*, *disagree* and *discuss* labels in FNC. In addition, lines 18-23 show the results when we discard the *neutral* label in SNLI.

Overall, the results show that using SNLI and domain adaptation technique does not have a significant positive impact on the FNC target test data. While it results in an improvement on the $F_1$ score for the *agree* and *disagree* labels, it significantly drops the performance for the *discuss* label. This is likely because the SNLI and FNC datasets are designed for different tasks and that there is a large difference in the data distribution between the two datasets; they cover different topics and the SNLI dataset consists of sentence to sentence relationships, as opposed to sentence to document relationships in FNC. However, there is a promising result in Table 4.8 when using BOW and domain adaptation with the *hierarchy* scheme and without using the SNLI *neutral* label. It outperforms all models including baselines in terms of $F_1$ score (line 19).

In summary, the results show:

- The best $F_1$ performance is achieved by our model when it uses the combination of BOW and the *hierarchy* scheme with domain adaptation. It slightly outperforms the baselines on $F_1$, especially on the *disagree* label class (see line 19).

- In general, the *source* SNLI data doesn't seem to improve the performance of our model for target FNC data through adversarial domain adaptation. This is probably because of the significant differences between the tasks, such as the differences in task purpose (inference vs stance detection), and distributional differences such as a difference in topics and a large difference in terms of average document length.

| | Model | Source | Target | Acc. | Acc. *supports/refutes* | Macro $F_1$ | $F_1$ *supports/refutes* |
|---|---|---|---|---|---|---|---|
| | **FEVER Article** | | | | | | |
| 1. | BOW | - | FEVER | 62.5 | 95.9 / 28.4 | 57.5 | 72.1 / 42.8 |
| 2. | CNN | - | FEVER | 61.6 | 97.1 / 25.3 | 55.7 | 71.9 / 39.5 |
| 3. | BOW + CNN | - | FEVER | 61.7 | 97.7 / 25.0 | 55.7 | 72.1 / 39.2 |
| 4. | BOW | SNLI | FEVER | 60.7 | 96.8 / 23.9 | 54.6 | 71.4 / 37.7 |
| 5. | [BOW + DA] | SNLI | FEVER | 61.8 | 93.9 / 29.1 | 57.2 | 71.3 / 43.0 |
| 6. | CNN | SNLI | FEVER | 59.4 | 96.7 / 21.2 | 52.4 | 70.6 / 34.1 |
| 7. | [CNN + DA] | SNLI | FEVER | 57.8 | 96.6 / 18.1 | 49.8 | 69.8 / 29.8 |
| 8. | BOW + CNN | SNLI | FEVER | 60.9 | 95.1 / 25.9 | 55.3 | 71.0 / 39.6 |
| 9. | TF + [CNN + DA] | SNLI | FEVER | 60.9 | 95.4 / 25.7 | 55.4 | 71.2 / 39.5 |
| | **FEVER Article + Balanced Labels** | | | | | | |
| 10. | BOW | - | FEVER | 65.2 | 82.2 / 47.9 | 64.1 | 70.5 / 57.7 |
| 11. | CNN | - | FEVER | 66.0 | 79.4 / 52.4 | 65.4 | 70.3 / 60.4 |
| 12. | BOW + CNN | - | FEVER | 66.8 | 80.2 / 53.1 | 66.1 | 70.9 / 61.2 |
| 13. | BOW | SNLI | FEVER | 64.4 | 79.3 / 49.1 | 63.5 | 69.2 / 57.7 |
| 14. | [BOW + DA] | SNLI | FEVER | 64.6 | 79.8 / 49.1 | 63.7 | 69.5 / 57.8 |
| 15. | CNN | SNLI | FEVER | 62.3 | 82.2 / 42.0 | 60.6 | 68.8 / 52.4 |
| 16. | [CNN + DA] | SNLI | FEVER | 61.1 | 84.2 / 37.5 | 58.7 | 68.6 / 48.8 |
| 17. | BOW + CNN | SNLI | FEVER | 63.3 | 69.8 / 56.6 | 63.1 | 65.8 / 60.4 |
| 18. | BOW + [CNN + DA] | SNLI | FEVER | 63.8 | 72.6 / 54.8 | 63.4 | 66.9 / 59.9 |
| | **FEVER Sentence + Balancing** | | | | | | |
| 19. | BOW | - | FEVER | 68.0 | 81.1 / 54.6 | 67.4 | 72.0 / 62.7 |
| 20. | CNN | - | FEVER | 66.8 | 78.2 / 55.2 | 66.4 | 70.6 / 62.1 |
| 21. | BOW + CNN | - | FEVER | 68.1 | 77.2 / 58.8 | 67.8 | 71.1 / 64.5 |
| 22. | BOW | SNLI | FEVER | 66.8 | 77.4 / 55.9 | 66.4 | 70.3 / 62.4 |
| 23. | [BOW + DA] | SNLI | FEVER | 65.4 | 80.8 / 49.6 | 64.5 | 70.4 / 58.6 |
| 24. | CNN | SNLI | FEVER | 64.8 | 82.7 / 46.5 | 63.6 | 70.5 / 56.6 |
| 25. | [CNN + DA] | SNLI | FEVER | 63.7 | 82.3 / 44.7 | 62.3 | 69.7 / 54.8 |
| 26. | BOW + CNN | SNLI | FEVER | 65.5 | 72.2 / 58.6 | 65.3 | 68.0 / 62.6 |
| 27. | BOW + [CNN + DA] | SNLI | FEVER | 65.8 | 72.8 / 58.7 | 65.6 | 68.4 / 62.8 |

Table 4.9: SNLI → FEVER. Results on the FEVER test data for models trained using FEVER training data and SNLI data. BOW, CNN and DA refer to our model when it uses bag of words features, convolutional features, and domain adaptation, respectively. When DA is present, square brackets indicate which features are passed to the domain adaptation component. The *hierarchy* in parentheses refers to our model with two level prediction scheme as explained in 4.3.3. We show the results of the models based on the smallest loss for validation set across 5 independent runs.

### 4.4.3 Performance Analysis: SNLI → FEVER

Table 4.9 shows the results of different models on the target FEVER test data. Similar to the previous section, lines 1-9 show the results when all train data is used, while lines 10-18 show the results when FEVER train data is balanced across classes. Since the FEVER dataset has more stance annotations at the sentence level (as compared to document level), we repeat our experiments at the sentence level (lines 19-27); note that the source SNLI data contains short length sentences. In these experiments, the *entailment* and *conflict* labels in source SNLI correspond to the *SUPPORTS* and *REFUTES* labels in target FEVER.

The results show that both balancing the labels (lines 10-18) and considering FEVER examples at the sentence level (lines 19-27) can significantly improve model performance; around 5% on Accuracy and Macro $F_1$ with balanced labels, and around 3% improvement with sentences instead of entire documents.

The results indicate that SNLI data is not able to improve the results on FEVER data through adversarial domain adaptation, and there is a negligible difference when using domain adaptation compared to not using it. The reason is, most likely, due to the difference between the different tasks that created the SNLI and FEVER datasets.

In summary, the results show:

- Balancing the FEVER train data can generally help the models to perform better (comparing lines 1-9 with lines 10-18), and the results at sentence level (lines 19-27) are much better than results at document level (lines 1-18).

- The best performance is achieved with the combination of BOW and CNN without domain adaptation and with balanced train data at sentence level (see line 21).

- In general, the *source* SNLI data can't improve the performance of our model for *target* FEVER data with adversarial domain adaptation, even at sentence level. This might be because of the considerable task difference between FEVER and SNLI.

| | Model | Source | Target | Acc. | Acc. *supports/refutes* | Macro $\mathbf{F}_1$ | $\mathbf{F}_1$ *supports/refutes* |
|---|---|---|---|---|---|---|---|
| | **FEVER Article** | | | | | | |
| 1. | BOW | - | FEVER | 62.5 | 95.9 / 28.4 | 57.5 | 72.1 / 42.8 |
| 2. | CNN | - | FEVER | 61.6 | 97.1 / 25.3 | 55.7 | 71.9 / 39.5 |
| 3. | BOW + CNN | - | FEVER | 61.7 | 97.7 / 25.0 | 55.7 | 72.1 / 39.2 |
| 4. | BOW | FNC | FEVER | 57.3 | 96.9 / 16.8 | 48.8 | 69.6 / 28.0 |
| 5. | [BOW + DA] | FNC | FEVER | 57.3 | 96.2 / 17.6 | 49.3 | 69.5 / 29.1 |
| 6. | CNN | FNC | FEVER | 56.8 | 99.1 / 13.6 | 46.9 | 69.9 / 23.9 |
| 7. | [CNN + DA] | FNC | FEVER | 56.8 | 97.9 / 14.9 | 47.5 | 69.6 / 25.4 |
| 8. | BOW + CNN | FNC | FEVER | 58.8 | 98.9 / 17.9 | 50.5 | 70.8 / 30.1 |
| 9. | TF + [CNN + DA] | FNC | FEVER | 56.6 | 95.7 / 16.7 | 48.4 | 69.0 / 27.8 |
| | **FEVER Article + Balanced Labels** | | | | | | |
| 10. | BOW | - | FEVER | 65.2 | 82.2 / 47.9 | 64.1 | 70.5 / 57.7 |
| 11. | CNN | - | FEVER | 66.0 | 79.4 / 52.4 | 65.4 | 70.3 / 60.4 |
| 12. | BOW + CNN | - | FEVER | 66.8 | 80.2 / 53.1 | 66.1 | 70.9 / 61.2 |
| 13. | BOW | FNC | FEVER | 59.8 | 71.0 / 48.4 | 59.3 | 64.1 / 54.4 |
| 14. | [BOW + DA] | FNC | FEVER | 60.6 | 73.1 / 47.9 | 59.9 | 65.2 / 54.6 |
| 15. | CNN | FNC | FEVER | 57.7 | 91.9 / 22.7 | 51.7 | 68.7 / 34.7 |
| 16. | [CNN + DA] | FNC | FEVER | 56.1 | 98.1 / 13.2 | 46.2 | 69.3 / 23.0 |
| 17. | BOW + CNN | FNC | FEVER | 57.8 | 94.1 / 20.7 | 51.1 | 69.3 / 32.8 |
| 18. | BOW + [CNN + DA] | FNC | FEVER | 50.5 | 100.0 / 0.0 | 33.6 | 67.1 / 0.0 |

Table 4.10: FNC → FEVER. Results on the FEVER test data for models trained using FEVER training data and FNC data. BOW, CNN and DA refer to our model when it uses bag of words features, convolutional features, and domain adaptation, respectively. When DA is present, square brackets indicate which features are passed to the domain adaptation component. We show the results of the models based on the smallest loss for validation set across 5 independent runs.

### 4.4.4 Performance Analysis: FNC → FEVER

Table 4.10 shows the results of different models on the target FEVER test data. Lines 1-9 show the results when all train data is used at each epoch for training, while lines 10-18 show the results when the FEVER train data is balanced across *SUPPORTS* and *REFUTES* labels. In these experiments, the *agree* and *disagree* labels in source FNC are mapped to the *SUPPORTS* and *REFUTES* labels in target FEVER data.

Overall, the results show that the source FNC is not able to improve the performance on the target FEVER data. None of the domain adaptation models outperform the BOW and CNN model trained only on FEVER data. The reasons are that the number of FNC *agree* and *disagree* labels is just a fraction of the size of the *SUPPORTS* and *REFUTES* labels in the FEVER data, and FNC data is imbalanced in terms of label frequency.

In summary, the results show:

- The best performance is achieved by the combination of BOW and CNN without domain adaptation and with balanced train data (see line 12).
- In general, the *source* FNC does not improve the performance of our models tested on *target* FEVER data. Possible reasons include the fact that the source FNC is much smaller than target FEVER data, and that the source FNC is different and unbalanced and may bias the model incorrectly.

### 4.4.5 Training Loss Trends

While the logic behind the adversarial domain adaptation model makes sense in theory, we sought to empirically confirm the impact of the domain adaptation technique on changing the classification and domain losses during training. To do this, we examine the losses of the best FEVER→FNC model, the BOW + [CNN + DA] with the *hierarchy* scheme, after each training epoch. The data is shown in Figure 4-3.

From the figure, we can see that both losses are unstable during the early epochs of training. This is because of the learning rate and gradient reversal constant schedules, which couple an initially high learning rate with an initially low gradient reversal

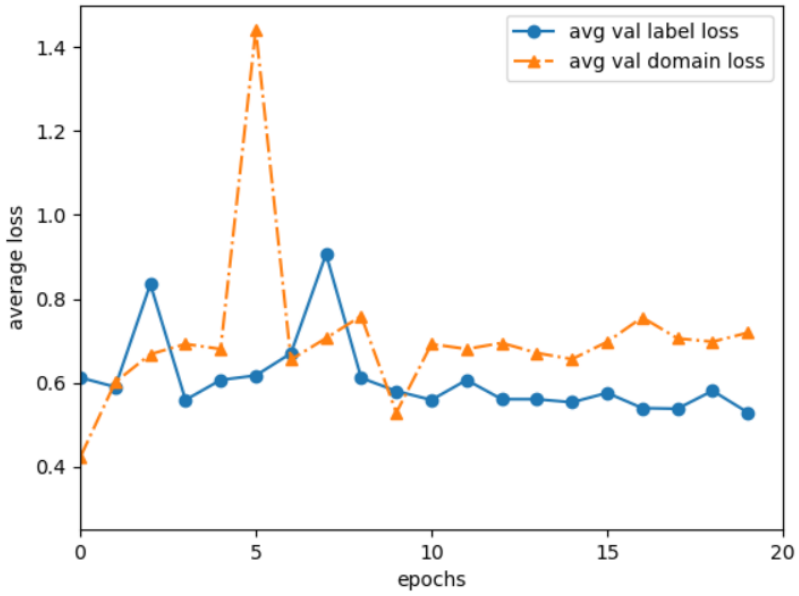


Figure 4-3: The classification and domain adaptation losses on validation data across training epochs for our best FEVER→FNC model; BOW + CNN + DA with the *hierarchy* scheme.

constant. Then, after around 10 epochs, the classification and domain adaptation losses plateau as training stabilizes. In general, the classification loss slowly decreases as the label prediction component in the model attempts to minimize its loss, while the domain adaptation loss slowly increases as the model attempts to maximize the domain loss so that it cannot distinguish between its source and target examples as we discussed in Section 4.2.3.

## 4.5 Summary

In summary, we demonstrated that using adversarial domain adaptation with our models (the BOW and CNN based models) could improve the stance detection task on the target data depending on matching between the target and source datasets in terms of data distribution, topics, desired tasks, etc. In particular, our domain adaptation model could outperform existing state of the art results for the FNC target data when using the source FEVER data, and the FEVER data was successfully used to supplement FNC which has a limited label data through domain adapta-

tion. We repeated our experiments with domain adaption using source SNLI data for target FNC, using source FNC data for target FEVER, and using source SNLI for target FEVER. The results of these experiments showed that the source data through domain adaptation couldn't significantly improve the performance on target data. Overall, our results show that adversarial domain adaptation can be used successfully for the stance detection task. However, it requires careful data selection and similar tasks.

# Chapter 5

# Stance Detection Demonstration

In this chapter, we introduce an automatic demonstration system created to detect the stance between an input claim and document pair. For our model, we use our best trained model (see Chapter 4) which was produced when (*i*) using the combination of TF and CNN models, (*ii*) using source data through the adversarial domain adaptation technique, and (*iii*) using the *hierarchy* scheme. This model outperforms the state of the art performance on the publicly available Fake News Challenge dataset. In the following sections, we will discuss in detail the architecture and design of our system.

## 5.1 Demonstration System Design

The demonstration system essentially has two sides; server and client. Our pretrained stance detection model is located at the server side. Figure 5-1 shows the high level design of the system. The user at the client side inputs a claim-document pair; there are not any restrictions on the contents of these inputs. Then, the client sends an HTTP request to the server for the model with the desired inputs to detect the stance between the input pair. The server then processes the inputs by tokenizing the text, removing stop words, and extracting the TF and TFIDF features. Then, these inputs are passed to our pretrained model, and the predicted output is sent back to the user. The output is a set of predicted scores in the *hierarchy* scheme for *related*

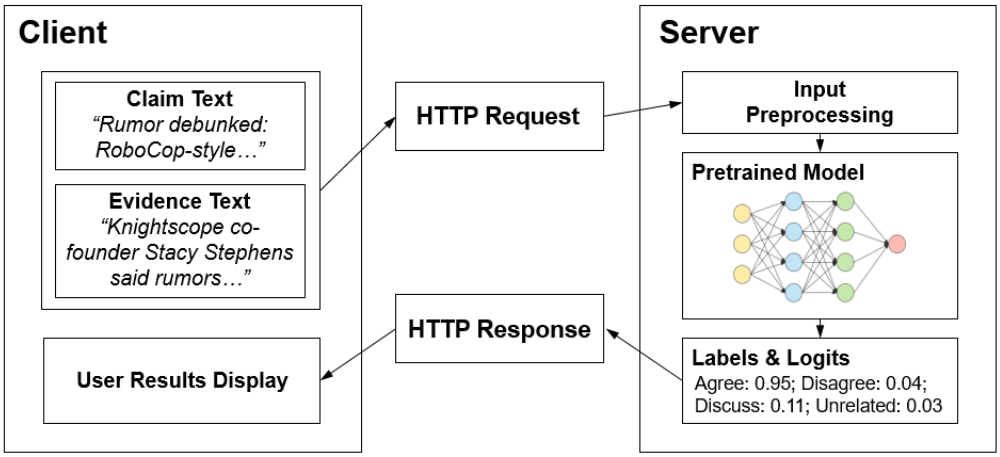


Figure 5-1: The high level design of the demonstration system. The client sends a claim-document pair to the server to be analyzed. Upon receiving the request, the server processes the input texts, gives them through our pretrained stance detection model, and then returns the results. The client receives these results and displays them to the user.

or *unrelated*, then for *agree*, *disagree* and *discuss*. Finally, to give a set of rationales for the model prediction, the server further processes the claim with each sentence in the document, and the stance scores for the sentences in the input document with respect to the claim are returned to the user. The relevant sentences in the document are highlighted and color coded corresponding to the stance labels.

## 5.2 An Example through the Demonstration System

Figures 5-2, 5-3, 5-4, and 5-5 are screenshots of our demonstration system when an example input is passed through the system. Figure 5-2 displays the input page in which the user can submit two text inputs as claim and document. Figure 5-3 displays the system predicted stance score at document level, and the sentences in the document that support the predictions are highlighted. Figure 5-4 and Figure 5-5 display the predicted stance score for each sentence in the document with respect to the given claim.

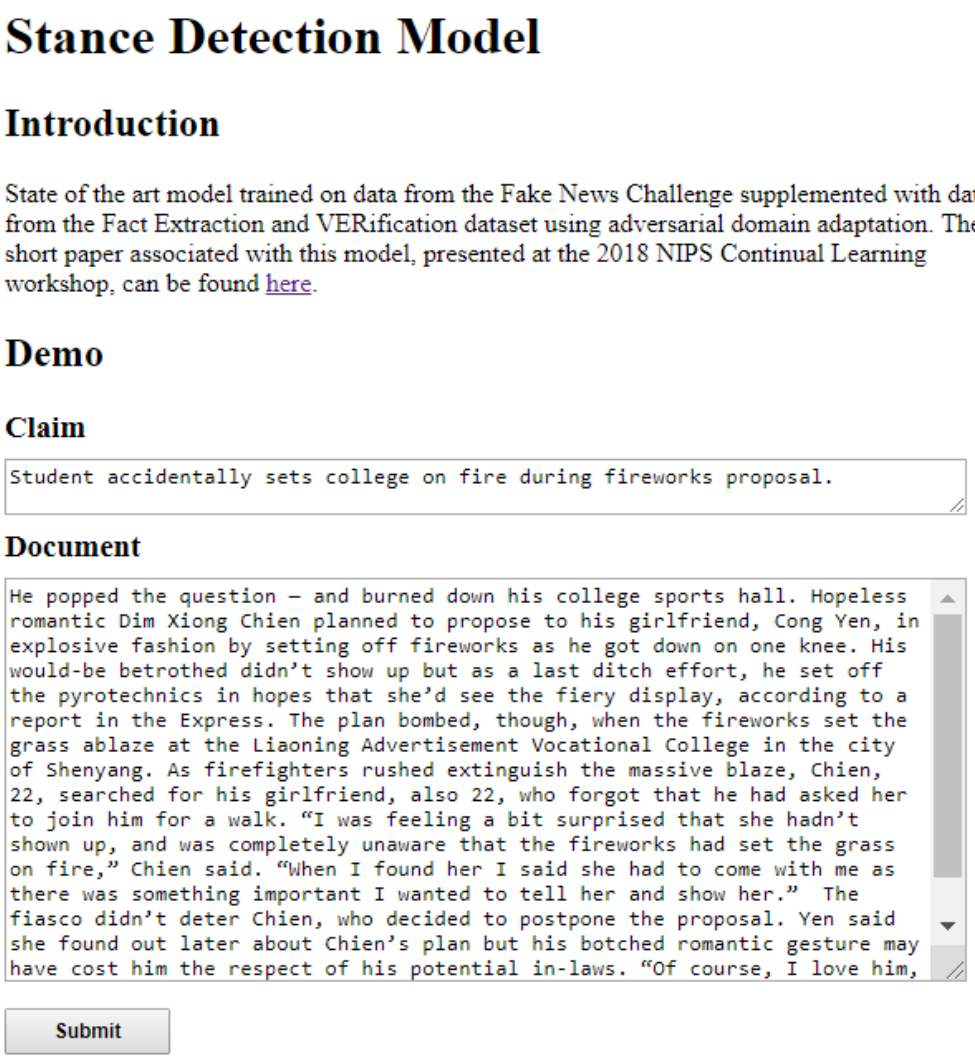


Figure 5-2: The input page of the demonstration system. The user can enter two text inputs as claim and document. Then, the inputs are processed by the demonstration system after clicking the submit button.

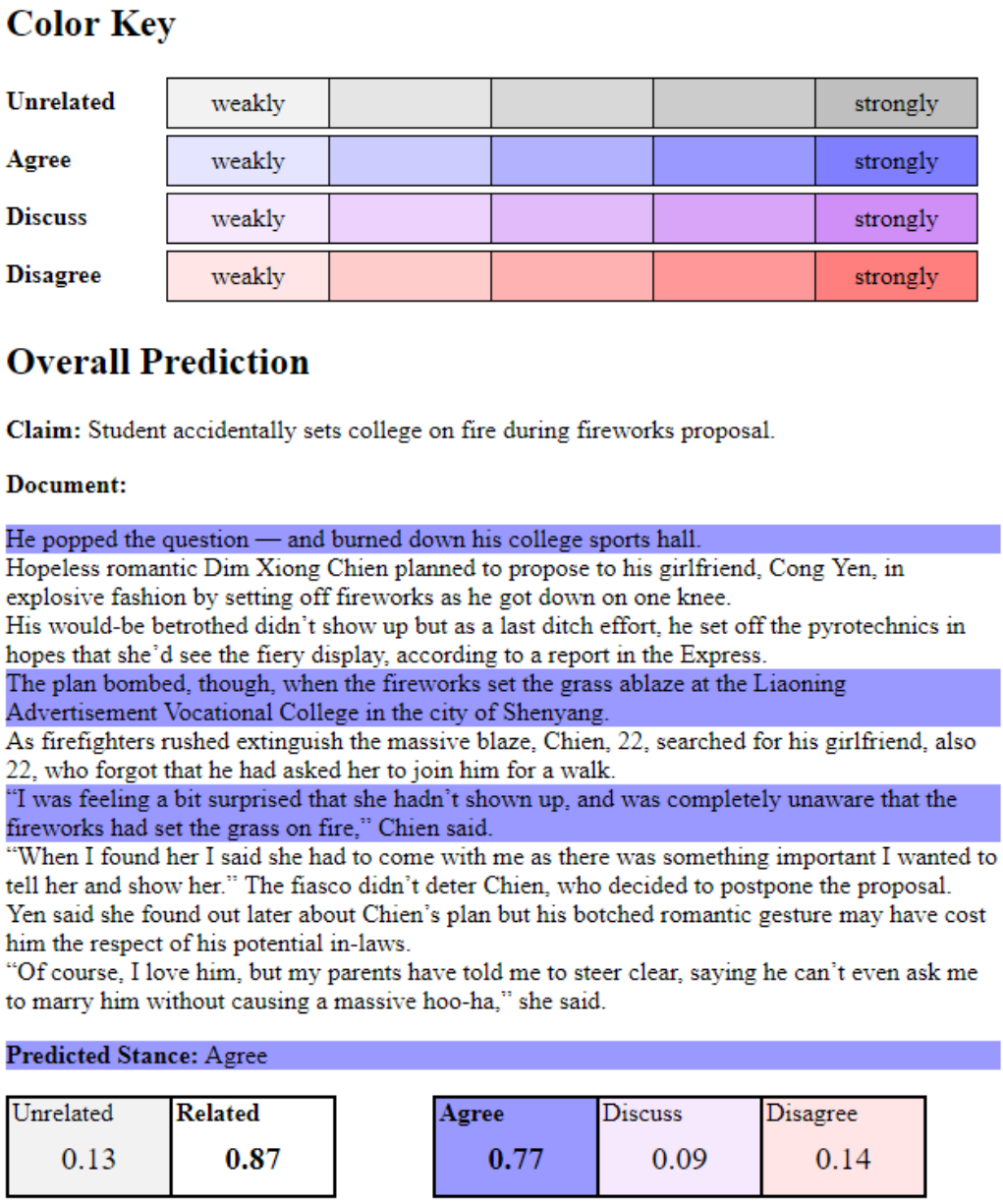


Figure 5-3: The first part of the results page contains a color key and statistics of our model predictions for the input claim-document pair at a holistic level. The sentences in the document that support the predictions are highlighted.

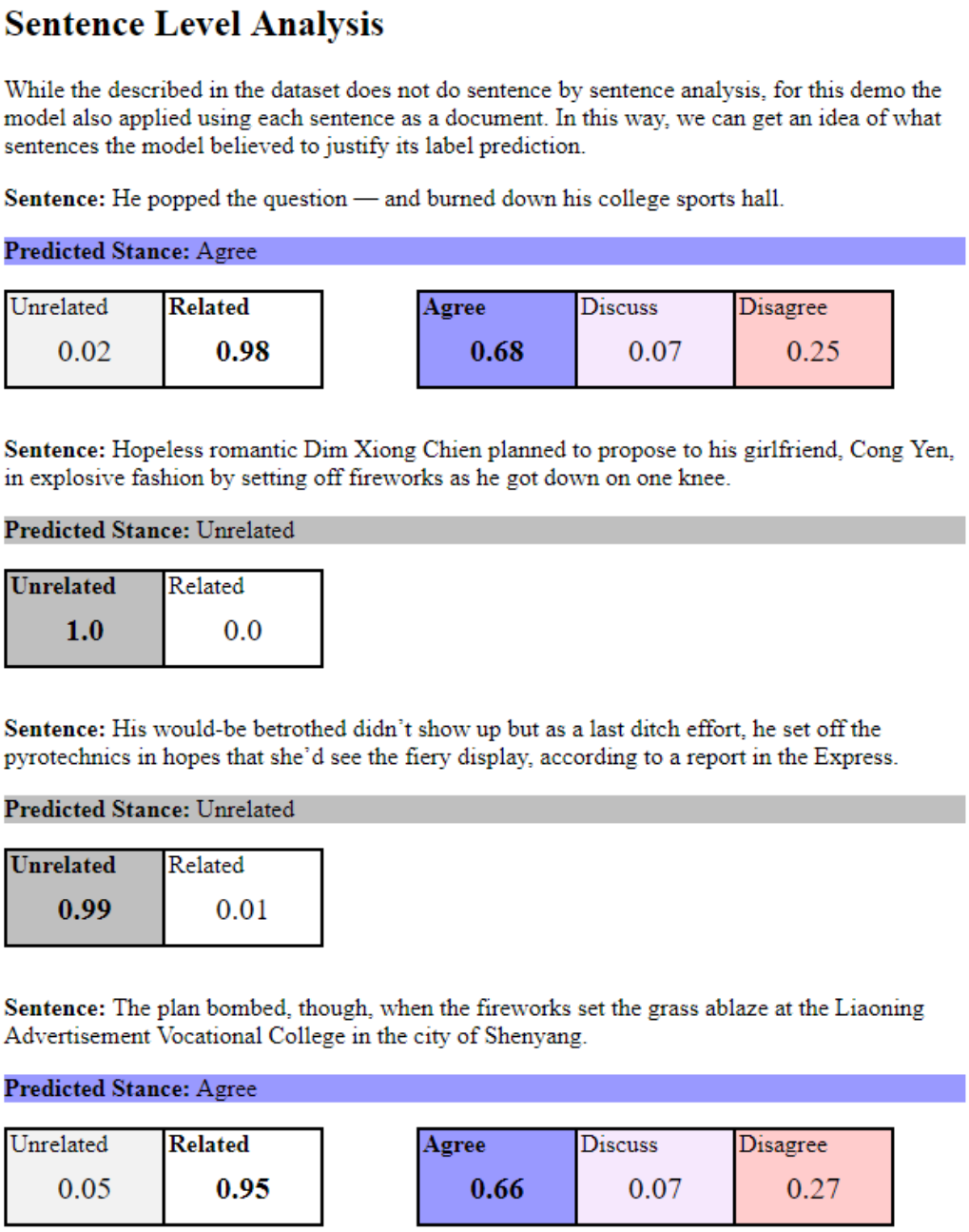


Figure 5-4: The second part of the results page contains model predictions specific to each sentence in the document. In particular, the specific sentence along with the claim is passed to the model and the model predictions are displayed.

**Sentence:** As firefighters rushed extinguish the massive blaze, Chien, 22, searched for his girlfriend, also 22, who forgot that he had asked her to join him for a walk.

**Predicted Stance:** Unrelated

| **Unrelated** | Related |
|---|---|
| **0.98** | 0.02 |

**Sentence:** "I was feeling a bit surprised that she hadn't shown up, and was completely unaware that the fireworks had set the grass on fire," Chien said.

**Predicted Stance:** Agree

| Unrelated | **Related** |
|---|---|
| 0.11 | **0.88** |

| **Agree** | Discuss | Disagree |
|---|---|---|
| **0.68** | 0.11 | 0.21 |

**Sentence:** "When I found her I said she had to come with me as there was something important I wanted to tell her and show her." The fiasco didn't deter Chien, who decided to postpone the proposal.

**Predicted Stance:** Unrelated

| **Unrelated** | Related |
|---|---|
| **1.0** | 0.0 |

**Sentence:** Yen said she found out later about Chien's plan but his botched romantic gesture may have cost him the respect of his potential in-laws.

**Predicted Stance:** Unrelated

| **Unrelated** | Related |
|---|---|
| **0.99** | 0.01 |

**Sentence:** "Of course, I love him, but my parents have told me to steer clear, saying he can't even ask me to marry him without causing a massive hoo-ha," she said.

**Predicted Stance:** Unrelated

| **Unrelated** | Related |
|---|---|
| **1.0** | 0.0 |

Figure 5-5: This figure is in continuous of the previous part of the system outputs at sentence level.

## 5.3 Technologies

Our pretrained model is created using Python 3 Tensorflow[1]. Our demonstration system uses the Flask[2] framework. For preprocessing the inputs, e.g., tokenizing and stop word removal steps, we use Keras[3] library functions. To extract BOW features, e.g., TF and TF-IDF), we use existing scikit-learn[4] library functions. Finally, the server is currently being run on a lab cluster CPU computer.

## 5.4 Extensibility

The current design is easily extensible and additional models, especially Python models, can be included to the system over time to demonstrate progress on the task.

When adding models to the demo, there are potentially some possible concerns include memory restrictions and prediction latency. If the model takes up too much memory, then the server process may freeze or otherwise fail. This would happen if the model has a very large amount of parameters to the point where the amount of memory required exceeds the RAM available for the server process. To resolve this issue, either the amount of memory that the model is fetching should be reduced or a machine with a larger memory capacity should be used. Next, the latency of the system is directly linked with how long it takes to load the saved model, process the inputs, and predict the outputs. Currently the model is fairly light weight. However, if the new added model is more computationally expensive, then the latency can significantly increase for the user. In this case, there is a trade-off between the speed (with using a simpler model) and performance (with using a complex model with high accuracy) of the system.

[1]https://www.tensorflow.org/
[2]http://flask.pocoo.org/
[3]https://keras.io/
[4]https://scikit-learn.org/

## 5.5 Summary

We presented a demonstration system to serve our best model to process arbitrary user determined input. Our current best model is based on using BOW and CNN features, *hierarchy* schemes, and domain adaptation that could achieve the state of the art result on a publicly available benchmark. This system is helpful in demonstrating this model for stance detection task. Furthermore, it is designed in an extensible manner which makes it easy to add future state of the art models.

# Chapter 6

# Conclusion

## 6.1 Summary

In this thesis, we presented neural and adversarial domain adaptation approaches to solve the problem of fact checking and the problem of stance detection. The primary contributions of this work are summarized as follows.

**Fact Checking:** The fact checking task involves predicting whether a given textual claim is factually true or not. To address this task, we presented improved CNN and LSTM models as well a novel stacked Bi-LSTM and CNN model. Our models outperformed several baselines including the state of the art. We performed our experiments using a publicly available dataset, LIAR.

**Stance Detection:** The stance detection task involves predicting the perspective of a document to an input claim. To address this task, we presented different configurations of BOW and CNN based models that use the adversarial domain adaptation technique. We performed our experiments using FNC, FEVER and SNLI datasets. We found that using source FEVER data for target FNC data improves model performance for the FNC task. This is likely because the source is much larger than the target data and both datasets are created for the same task, allowing for useful information transfer.

**Stance Detection Demonstration System:** Finally, we created a demonstration system to help users understand the current capabilities of presented models. In particular, the user is able to enter arbitrary textual claim and document and get the model predictions at document and sentence level. This helps to demonstrate the strengths and shortcomings of the models, and it can be useful in benchmarking for future progress in the field.

## 6.2 Future Work

The ultimate goal of fact checking and stance detection tasks is to create a system which can alert users about misleading information or fake news by presenting enough evidence. The fact checking task directly addresses this problem by outputting the final decision on an input claim as to whether it is factually true or not. The stance detection task outputs the stances of documents against claims; these outputs can be aggregated and used as evidence for the fact checking decision.

For the fact checking task, as examined in this thesis, one major issue is the lack of labeled data. This is while neural models need sufficient data to be trained. So far, most of the fact checking datasets (e.g., LIAR dataset) are very limited in size, with most of topics only having a handful of claims and with most of the speakers only having a single claim. These challenges make it hard to train a model on such datasets to achieve a good result. Therefore, it is important to develop more labeled data for this task.

The stance detection task also suffers from dataset size problems. Most of the existing datasets for this task have undesirable biases, due to the way that they are created. For example, the FNC dataset has around 100k claims, however often the same documents are associated many times to different claims. This means that the dataset is not as diverse as the size would suggest. Furthermore, the FNC dataset is very imbalanced in terms of label frequency, which de facto limits the performance of the models over all stance labels. Additional carefully constructed datasets would be very useful to help advance research on this problem.

To tackle the dataset size limitation issue, we applied adversarial domain adaptation in order to use data from similar datasets, e.g. FEVER and SNLI, to supplement each other. We achieved state of the art performance using source FEVER data on target FNC data with this method, and this method could possibly be applied for other similar datasets. While we used the adversarial domain adaptation technique for our work, other domain adaptation techniques can be also investigated to see if they are more effective. Finally, any future datasets that are related to the stance detection task can be tested using domain adaptation to potentially improve existing tasks.

# Bibliography

[Augenstein et al., 2016] Augenstein, I., Rocktäschel, T., Vlachos, A., and Bontcheva, K. (2016). Stance detection with bidirectional conditional encoding. *CoRR*, abs/1606.05464.

[Baird et al., 2017] Baird, S., Sibley, D., and Pan, Y. (2017). Talos targets disinformation with fake news challenge victory. In *Fake News Challenge*.

[Baly et al., 2018] Baly, R., Mohtarami, M., Glass, J., Màrquez, L., Moschitti, A., and Nakov, P. (2018). Integrating stance detection and fact checking in a unified corpus. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics*, New Orleans, LA, USA.

[Bizer et al., 2009] Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., and Hellmann, S. (2009). Dbpedia - a crystallization point for the web of data. *Web Semant.*, 7(3):154–165.

[Bouma, 2009] Bouma, G. (2009). Normalized (pointwise) mutual information in collocation extraction. In *From Form to Meaning: Processing Texts Automatically, Proceedings of the Biennial GSCL Conference 2009*, volume Normalized, pages 31–40, Tübingen.

[Bowman et al., 2015] Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

[Chen, 2016] Chen, G. (2016). A gentle tutorial of recurrent neural network with error backpropagation. *CoRR*, abs/1610.02583.

[Ciampaglia et al., 2015] Ciampaglia, G. L., Shiralkar, P., Rocha, L. M., Bollen, J., Menczer, F., and Flammini, A. (2015). Computational fact checking from knowledge networks. *CoRR*, abs/1501.03471.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

[Ferreira and Vlachos, 2016] Ferreira, W. and Vlachos, A. (2016). Emergent: a novel data-set for stance classification. In *HLT-NAACL*.

[Ganin and Lempitsky, 2014] Ganin, Y. and Lempitsky, V. (2014). Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*.

[Gong et al., 2013] Gong, B., Grauman, K., and Sha, F. (2013). Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 222–230.

[Graves and Schmidhuber, 2005] Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602 – 610. IJCNN 2005.

[Hanselowski et al., 2017] Hanselowski, A., PVS, A., Schiller, B., and Caspelherr, F. (2017). Team Athene on the fake news challenge. https://medium.com/@andre134679/team-athene-on-the-fake-news-challenge-28a5cf5e017b.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[Karadzhov et al., 2017a] Karadzhov, G., Gencheva, P., Nakov, P., and Koychev, I. (2017a). We built a fake news & click-bait filter: What happened next will blow your mind! In *Proceedings of the 2017 International Conference on Recent Advances in Natural Language Processing*, RANLP '17, Varna, Bulgaria.

[Karadzhov et al., 2017b] Karadzhov, G., Nakov, P., Màrquez, L., Barrón-Cedeño, A., and Koychev, I. (2017b). Fully automated fact checking using external sources. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 344–353. INCOMA Ltd.

[Karadzhov et al., 2017c] Karadzhov, G., Nakov, P., Màrquez, L., Barrón-Cedeño, A., and Koychev, I. (2017c). Fully automated fact checking using external sources. *CoRR*, abs/1710.00341.

[Kim, 2014] Kim, Y. (2014). Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882.

[Liu et al., 2017] Liu, P., Qiu, X., and Huang, X. (2017). Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742*.

[Mihaylov et al., 2015] Mihaylov, T., Georgiev, G., and Nakov, P. (2015). Finding opinion manipulation trolls in news community forums. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 310–314. Association for Computational Linguistics.

[Mihaylov and Nakov, 2016] Mihaylov, T. and Nakov, P. (2016). Hunting for troll comments in news community forums. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 399–405, Berlin, Germany.

[Mihaylova et al., 2018] Mihaylova, T., Nakov, P., Marquez, L., Barron-Cedeno, A., Mohtarami, M., Karadzhov, G., and Glass, J. (2018). Fact checking in community forums. In *Proceedings of AAAI*, New Orleans, LA, USA.

[Mikolov et al., 2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

[Mikolov et al., 2013b] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed Representations of Words and Phrases and their Compositionality. *CoRR*, abs/1310.4546.

[Mohammad et al., 2016] Mohammad, S., Kiritchenko, S., Sobhani, P., Zhu, X., and Cherry, C. (2016). Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41, San Diego, California. Association for Computational Linguistics.

[Mohtarami et al., 2018] Mohtarami, M., Baly, R., Glass, J., Nakov, P., Màrquez, L., and Moschitti, A. (2018). Automatic stance detection using end-to-end memory networks. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL-HLT '18, New Orleans, LA, USA.

[Mohtarami et al., 2016] Mohtarami, M., Belinkov, Y., Hsu, W.-N., Zhang, Y., Lei, T., Bar, K., Cyphers, S., and Glass, J. (2016). Sls at semeval-2016 task 3: Neural-based approaches for ranking in community question answering. In *Proceedings of NAACL-HLT Workshop on Semantic Evaluation*, pages 753–760, San Diego, California. Association for Computational Linguistics.

[Pan et al., 2011] Pan, S. J., Tsang, I. W., Kwok, J. T., and Yang, Q. (2011). Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[Pomerleau and Rao, 2017] Pomerleau, D. and Rao, D. (2017). Fake news challenge.

[Ramos et al., 2003] Ramos, J. et al. (2003). Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142.

[Riedel et al., 2017] Riedel, B., Augenstein, I., Spithourakis, G. P., and Riedel, S. (2017). A simple but tough-to-beat baseline for the fake news challenge stance detection task. *CoRR*, abs/1707.03264.

[Severyn and Moschitti, 2015] Severyn, A. and Moschitti, A. (2015). Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of*

*the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382. ACM.

[Sukhbaatar et al., 2015] Sukhbaatar, S., Weston, J., Fergus, R., et al. (2015). End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

[Sun et al., 2017] Sun, S., Zhang, B., Xie, L., and Zhang, Y. (2017). An unsupervised deep domain adaptation approach for robust speech recognition. *Neurocomputing*, 257:79–87.

[Tan et al., 2016] Tan, M., dos Santos, C. N., Xiang, B., and Zhou, B. (2016). Improved representation learning for question answer matching. In *ACL (1)*.

[Thorne et al., 2018] Thorne, J., Vlachos, A., Christodoulopoulos, C., and Mittal, A. (2018). Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819. Association for Computational Linguistics.

[Vlachos and Riedel, 2014] Vlachos, A. and Riedel, S. (2014). Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 18–22, Baltimore, MD, USA.

[Vosoughi et al., 2018] Vosoughi, S., Roy, D., and Aral, S. (2018). The spread of true and false news online. *Science*, 359(6380):1146–1151.

[Wang, 2017] Wang, W. Y. (2017). "liar, liar pants on fire": A new benchmark dataset for fake news detection. *CoRR*, abs/1705.00648.

[Xu et al., 2018] Xu, B., Mohtarami, M., and Glass, J. (2018). Adversarial doman adaptation for stance detection. In *Proceedings of the Thirty-second Annual Conference on Neural Information Processing Systems (NIPS)–Continual Learning*.

[Yu et al., 2014] Yu, L., Hermann, K. M., Blunsom, P., and Pulman, S. (2014). Deep learning for answer sentence selection. *CoRR*, abs/1412.1632.

[Zarrella and Marsh, 2016] Zarrella, G. and Marsh, A. (2016). MITRE at semeval-2016 task 6: Transfer learning for stance detection. *CoRR*, abs/1606.03784.