**INTRODUCTION**

Fake news are online stories that appear to be factual but are not. They may appear in websites that appear to be legitimate, although often they are at websites with little or no real news. Unfortunately, sometimes fake news stories are picked up by legitimate media outlets and appear on their websites.The spread of fake news articles has generated noticeable concern recently, as false or misleading stories can spread faster and reach a wider audience over social media. Given that changing technology has played a major role in enabling the spread of fake news, a natural question is whether technology can also help warn users about false claims. While technology is certainly not advanced enough to evaluate the truth of a claim on its own, it could be used to aid journalists and make it easier for them to detect and debunk false statements.

While automated fact checking and stance detection has not yet garnered much attention from researchers, previous research in Natural Language Inference has worked on problems that are very similar to ours. NLI attempts to identify the relationship between two statements, by identifying whether two bodies of text support, contradict, or are neutral towards one another. Researchers in NLI have achieved reasonable success on this task using neural network models, and almost all of the the best performing models on the benchmark Stanford SNLI corpus have incorporated neural networks. Matching LSTMs were successfully used for NLI by [1] to achieve a performance of 86.1% on the SNLI dataset. tests a LSTM model, attention model, and a word-by-word attention model on the SNLI corpus. In their paper, word-by-word attention achieves the highest test accuracy of 83.5% on the SNLI dataset. A sequential LSTM-based model combined with a syntactic parsing model was used by to achieve 88.3% accuracy on the SNLI corpus.

With the advent of fake news being used to influence elections, the identification of false information has become an important task. Governments, newspapers and social media platforms are working hard on distinguishing credible news from fake news. The goal of the Fake News Challenge is to automate the process of identifying fake news by using machine learning and natural language processing. This process can be broken down into several stages. A first helpful step towards the identification of fake news is to understand what other news sources are saying about the same topic. That is why the fake news challenge initially focuses on stance detection. Stance detection comprises the estimation of the relative perspectives of two different text pieces on the same topic as described by. Specifically, the task is to estimate the stance of a news headline, relative to the contents of a news article which can but does not have to address the same topic. Thus, the relative stance of each headline-article pair has to be classified as either unrelated, discuss, agree or disagree. The discovery of a disagreeing headline-article pair does not necessarily correspond to the discovery of a fake article, but it is an automated first step which could make human reviewers aware of a discrepancy.

**RELEVANT PROBLEMS WITH FAKE NEWS**

It is said that " False travels around the whole world while truth is about to put on shoes"**.** With many individuals relying on the internet for daily news, fake news continues to be circulated on search engines and social media, which leads to inaccurate stories being virally shared worldwide. All of these fake articles being circulated contains outrageous headlines which were meant to attract the greatest amount of engagement from users, as well as pretending to be legitimate to gain credibility — at least at first glance. This led to exponential engagement to millions of users through mindless sharing on social media such as Facebook. Social media has surged exponentially in last decade. The growing internet and smartphone users have given birth to new threat of FAKE NEWS on the social media. People spreading and blindly believing the same is disastrous.

Social media has become an integral part of modern society as all are very keen to increase their social footprints and make their presence felt in the virtual communities and networks. But suddenly the very primary cause of this social media like creation and sharing of information, news, trends, best practices and opinions are being sidetracked as the menace of fake news is popping up its head out. The reasons for the fake news gaining the prominence are:

1. Anti-social elements of the society are purposely spreading such fake news due to some vested interest and gaining benefits.

2. Some users who are blind followers of important persons like celebrities or political figures often don’t take pain to check the authenticity of news and blindly do like/comment/share.

3. The present form of IT Act is not equipped with proper provisions to check the spreading of fake news.

4. Also the growing smart phones penetration can be hold responsible as it is extending the free access to social media through internet connectivity it offers.

**PROBLEM STATEMENT**

The spread of fake news articles has generated noticeable concern recently, as false or misleading stories can spread faster and reach a wider audience over social media. Given that changing technology has played a major role in enabling the spread of fake news, a natural question is whether technology can also help warn users about false claims. While technology is certainly not advanced enough to evaluate the truth of a claim on its own, it could be used to aid journalists and make it easier for them to detect and debunk false statements.

There are several applications which have been designed for this purpose.Our application will explore the use Natural Language Processing techniques to determine whether a body of text agrees, disagrees, discusses, or is unrelated to another. This model could be applied as an automatic fact checker that could read an article, and then find other articles that either disagree or agree with its content. For this we will use methods from the field of Natural Language Inference to build a model that classifies the relationship between a news article headline and the body of a different news article.

With the advent of fake news being used to influence elections, the identification of false information has become an important task. Governments, newspapers and social media platforms are working hard on distinguishing credible news from fake news. And for this people developed the Fake News Challenge with the goal to automate the process of identifying fake news by using machine learning and natural language processing. This process can be broken down into several stages. A first helpful step towards the identification of fake news is to understand what other news sources are saying about the same topic. This is a problem which needs to be dealt with so we have also applied some of the concepts of deep learning to detect fake news and stance.

**SOLUTION AND APPROACH**

Our solution approach is mainly based on the key fact that through our proposed idea we are able to find out a key relation between the given headline and its corresponding body text. Hand-crafted features of headline-body pair, including bag of words and n-grams matching, have already been used to achieve moderate accuracy for stance classification and now recently neural-based encoder architectures are also been introduced to classify the problem. So basically our approach to identify the fake news can be categorised under various models.

We train 3 different models: a Bag of Vectors Model, LSTM, and RNN with Attention

**1.Baseline Model: Bag of Vectors**

The baseline model utilizes a straightforward bag of vectors approach. The model creates a L2-normalized sum of the embedding vectors for each of the words in the headline and body text. This new vector naively captures the meaning between the texts through summing their embedding vectors. To determine the prediction of the relationships between the headline and the body, the result is passed through a multilayer perceptron and softmax classifier to generate the final output. The intent of this model was to have a working baseline that has shown success in Natural Language Inference applications. Several MLP architectures with both tanh and relu activations were constructed for training.

**2. LSTMs**

The LSTM model is a sequence-to-sequence model replicated from and modified to our task. It uses two LSTM encoders to generate separate encodings of the headline and body text of dimension d. Next, the encodings are concatenated to form a vector of dimension 2d. Like the baseline model, the concatenated vector is passed through a multilayer perceptron and finally a softmax classifier to generate the final output. However, we modify this model from the original by treating the dimensions as hyperparameters, especially since our headlines and bodies are of significantly different lengths. We experiment by allowing the LSTM processing body text to be a much higher dimension than 100d, hoping that it will allow a better representation of the longer body.

**3. RNN with Attention**

An often used extension to the sequence-to-sequence LSTM model mentioned above is to add an attention mechanism that allows the body text to attend to the LSTM output layer from the headline to make the final prediction.

We also test multi-layer LSTMs with attention. Here the attention mechanism acts on the LSTM output in essentially the same manner, but the input article headlines and bodies pass through multiple layers of LSTMs before they are output. This may help us learn higher level structures in the text, at the cost of more parameters and more potential overfitting.

**Overall architecture with component description and dependency details**

To have things working smoothly and efficiently our system should be having the following requirements satisfied along with the mentioned softwares installed in it:

**Cudnn**

NVIDIA cuDNN is a GPU-accelerated library of primitives for deep neural networks(DNNs). It provides tuned implementations of routines that arise frequently in DNN applications, such as convultion, pooling, softmax, neuron activations.

cuDNN features customizable data layouts, supporting flexible dimension ordering, striding and subregions for the 4D tensors used as inputs and outputs to all of its routines. This flexibility allows easy integration into any neural net implementation and avoids the input/output transposition steps sometimes necessary with GEMM-based convolutions.

cuDNN is thread safe, and offers a context-based API that allows for easy multithreading and (optional) interoperability with CUDA streams. cuDNN allows DNN developers to easily harness state-of-the-art performance and focus on their application and the machine learning questions, without having to write custom code. When a developer leverages cuDNN, they can rest assured of reliable high performance on current and future NVIDIA GPUs, and benefit from new GPU features and capabilities in the future.

The cuDNN library is targeted at developers of DNN frameworks (eg. CAFFE, Torch). However it is easy to use directly and you do not need to know CUDA in order to use it. The example code below shows how to allocate storage for an input batch of images and a convolutional filter in cuDNN, and how to run the batch in the forward direction through a convolutional layer.

**Cuda**

CUDA is a [parallel computing](https://en.wikipedia.org/wiki/Parallel_computing) platform and [application programming interface](https://en.wikipedia.org/wiki/Application_programming_interface) (API) model created by [Nvidia](https://en.wikipedia.org/wiki/Nvidia). It allows [software developers](https://en.wikipedia.org/wiki/Software_developer) and [software engineers](https://en.wikipedia.org/wiki/Software_engineer) to use a CUDA-enabled [graphics processing unit](https://en.wikipedia.org/wiki/Graphics_processing_unit) (GPU) for general purpose processing – an approach termed [GPGPU](https://en.wikipedia.org/wiki/GPGPU) (General-Purpose computing on Graphics Processing Units). The CUDA platform is a software layer that gives direct access to the GPU's virtual [instruction set](https://en.wikipedia.org/wiki/Instruction_set) and parallel computational elements, for the execution of [compute kernels](https://en.wikipedia.org/wiki/Compute_kernel).

The CUDA platform is designed to work with programming languages such as [C](https://en.wikipedia.org/wiki/C_(programming_language)), [C++](https://en.wikipedia.org/wiki/C%2B%2B), and [Fortran](https://en.wikipedia.org/wiki/Fortran). This accessibility makes it easier for specialists in parallel programming to use GPU resources, in contrast to prior APIs like [Direct3D](https://en.wikipedia.org/wiki/Direct3D) and [OpenGL](https://en.wikipedia.org/wiki/OpenGL), which required advanced skills in graphics programming. Also, CUDA supports programming frameworks such as [OpenACC](https://en.wikipedia.org/wiki/OpenACC) and [OpenCL](https://en.wikipedia.org/wiki/OpenCL).

CUDA has several advantages over traditional general-purpose computation on GPUs (GPGPU) using graphics APIs:

· Scattered reads – code can read from arbitrary addresses in memory

· Unified virtual memory (CUDA 4.0 and above)

· Unified memory (CUDA 6.0 and above)

· [Shared memory](https://en.wikipedia.org/wiki/Shared_memory_(interprocess_communication)) – CUDA exposes a fast [shared memory](https://en.wikipedia.org/wiki/Scratchpad_RAM) region that can be shared among threads. This can be used as a user-managed cache, enabling higher bandwidth than is possible using texture lookups.[[14]](https://en.wikipedia.org/wiki/CUDA#cite_note-14)

· Faster downloads and readbacks to and from the GPU

· Full support for integer and bitwise operations, including integer texture lookups

**TensorFlow**

TensorFlow is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.

TensorFlow provides a [Python API](https://www.tensorflow.org/api_docs/python/), as well as [C++](https://www.tensorflow.org/api_docs/cc/), [Haskell](https://github.com/tensorflow/haskell), [Java](https://www.tensorflow.org/api_docs/java/reference/org/tensorflow/package-summary), [Go](https://godoc.org/github.com/tensorflow/tensorflow/tensorflow/go), and [Rust](https://github.com/tensorflow/rust) APIs. Third party packages are available for [C#](https://github.com/migueldeicaza/TensorFlowSharp), [Julia](https://github.com/malmaud/TensorFlow.jl), [R](https://github.com/rstudio/tensorflow), and [Scala](https://github.com/eaplatanios/tensorflow_scala). Among the applications for which TensorFlow is the foundation, are automated image captioning software, such as [DeepDream](https://en.wikipedia.org/wiki/DeepDream). RankBrain now handles a substantial number of search queries, replacing and supplementing traditional static algorithm based search results.