January 2018

Fact or Fiction

Anqi Lu
Worcester Polytechnic Institute

Charles J. Lovering
Worcester Polytechnic Institute

Cuong Tri Nguyen Dinh
Worcester Polytechnic Institute

Huyen Bui Thanh Nguyen
Worcester Polytechnic Institute

Fact or Fiction

A Major Qualifying Project
B-Term 2017
Microsoft New England Research and Development Center (NERD), Cambridge, Massachusetts,
submitted to the faculty of WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Bachelor of Science.

By:
Charles Lovering, Anqi Lu, Cuong Nguyen, Huyen Nguyen

Sponsor:
Microsoft Corporation

Liaison:
Ben Fersenheim

Mentor:
David Hurley

Project Advisor:
Professor Emmanuel Agu

Submitted on:
January 2nd, 2018

Abstract

Fake news is increasingly pervasive, and we address its problematic aspects to help people intelligently consume news. In this project, we research machine learning models to extract objective sentences, encouraging unbiased discussions based on facts. The most accurate model, a convolutional neural network, achieves an accuracy of 85.69%. The team implemented an end-to-end web system that highlights objective sentences in user input to make our model publicly accessible. The system also provides additional information about user input, such as links to related web pages. We evaluate our system both qualitatively by interviewing users, and quantitatively with surveys consisting of rating scale questions. Received positive feedback indicates the usability of our platform.

Acknowledgements

We would like thank our sponsor, Microsoft New England Research and Development Center (NERD) in Cambridge, Massachusetts for providing us with this wonderful opportunity. In particular, we thank David Hurley for making the prototype of this project and for his continuous effort in helping us with the project. We thank our MQP liaison, Ben Fersenheim for orchestrating the project between Microsoft and WPI, and providing all the resources we needed. Moreover, we are thankful of Professor Emmanuel Agu for his guidance during the PQP and MQP processes. Finally, we would like to thank everyone who participated in our user experience study for invaluable feedbacks.

List of acronyms

API	Application Programming Interface
CNN	Convolutional Neural Network
CNTK	Microsoft Computational Network Toolkit
HTTP	Hypertext Transfer Protocol
LSTM	Long Short Term Memory
SVM	Support Vector Machine
MQP	Major Qualifying Project
MVC	Model View Controller
MS	Microsoft
NLTK	Natural Language Toolkit
POS	Part of Speech
Tf-idf	Term frequency–inverse document frequency
RNN	Recurrent Neural Network
SDLC	Software Development Life Cycle
SUPR-Q	Standardized User Experience Percentile Rank Questionnaire
UI	User Interface
WPI	Worcester Polytechnic Institute

Chapter 1. Introduction

This project focuses on automated methods to distinguish objective statements from subjective ones, a specific case of distinguishing fact from fiction. In our project, “fact” is a piece of information presented as having objective reality, as defined by Merriam Webster. They are used in definitions, arguments, and generalities. For example, the sentence "Dungeons and Dragons is a game that involves storytelling and role-playing." is a fact. “Fiction”, or opinion, refers to subjective statements. They could be based on fact, but are hinged upon emotion or experience, or express an opinion. For example, "I think Dungeons and Dragons corrupts children’s minds." is an opinion. In short, a sentence is objective if the primary intention of a sentence is objective presentation of material that is factual to the reporter (Wiebe, 2001).

Distilling a document into facts and opinions is a difficult but useful capability. According to (Office of the Director of National Intelligence, 2017) and (Gray, 2017), there is a rise of propaganda termed Fake News, where subjective or inaccurate information were conveyed as facts, both in Europe and America. In addition, (Armstrong, 2016) cites research conducted by BuzzFeed using Buzzsumo, and comments that fake news had more engagement by Facebook users than mainstream stories between August 2016 and the election day as shown in Figure 1.

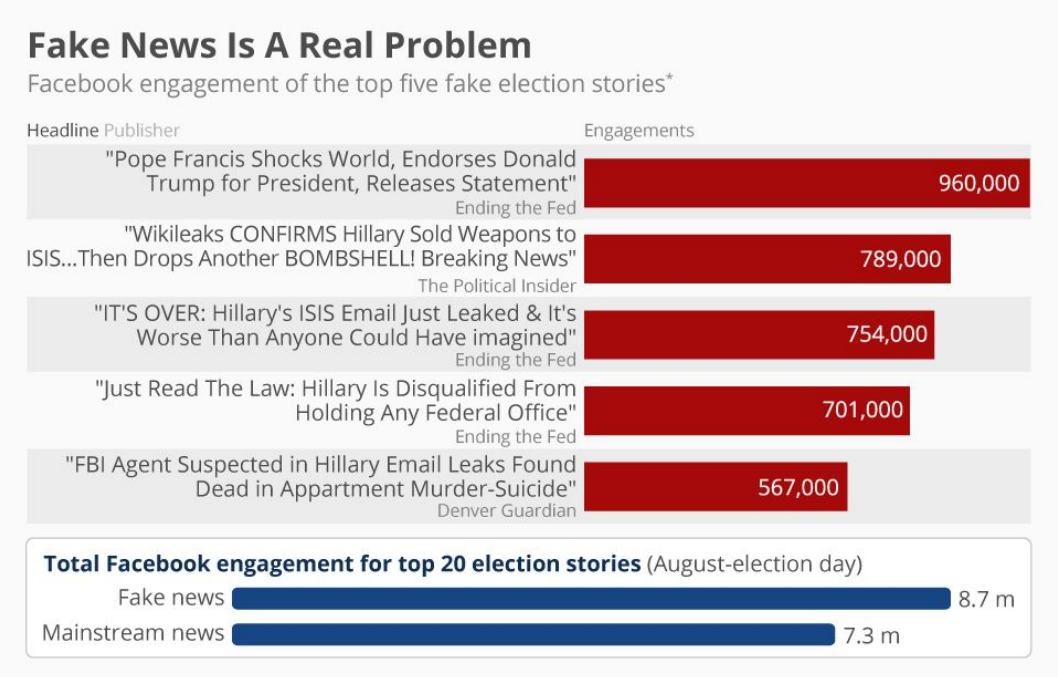

Figure 1. Facebook Engagement of the Top Five Fake Election Stories (source: Buzzsumo via Buzzfeed).

Therefore, being able to autonomously differentiate facts and opinions, and presenting this to users directly would help readers be more cognizant of polarized material. According to (Lelkes, 2016), due to rising polarization of a large variety of issues, inflammatory content is attractive as it disregards the opposing, disparate ideological view. Breaking down documents into facts, fact-checking them, and indicating which statements are opinion could help readers discern what is factual and encourage critical thinking. Moreover, this can foster conversations that are based on true facts rather than hear-says and opinions. This capacity is also beneficial from a purely technical standpoint. Being able to strip out facts or opinions from a document is a prerequisite for opinion mining and sentiment analysis systems (Pang & Lee, 2008), where the subjective sentences in a piece of text have to be extracted before the rest of the system can determine whether the opinion in the text is positive or negative.

1.1 Goals of our Project

The goal of this project is to encourage unbiased and constructive discussion by creating a platform to extract objective sentences from text entries using modern machine learning technologies.

The objectives of the project are as follows:

1. Experiment and train machine learning models to classify input statements as objective or subjective.
2. Deploy the classification model by implementing it as a scalable real-time machine learning web service.
3. Make the machine learning service available through a web platform that facilitates friendly user interaction.
4. Conduct user experience interviews to evaluate the usability of the end-to-end system.

1.2 Project Origin

This project began as part of Microsoft's Garage program, which is an internal hack and creativity competition that is meant to encourage Microsoft developers to work on projects of personal interest and thus fuel the growth of individual interests and research. As a submission to an annual hackathon, which is part of the Garage program, David Hurley and associates developed an initial prototype for the objective vs. subjective sentence classification system, which our work improves on.

1.3 Motivation

1.3.1 Fake News

In a report conducted by reports from U.S. News (2017), fake news was defined as an "ambiguous term for a type of journalism which fabricates materials to perpetuate a preferred narrative suitable for political, financial, or attention-seeking gain." More specifically, Allcott and Gentzkow's study (2017) categorizes fake news into six categories:

1. Unintentional reporting mistakes.
2. Rumors that do not originate from a particular news article.
3. Conspiracy theories.
4. Satire that is unlikely to be misconstrued as factual.
5. False statements by politicians.
6. Reports that are slanted or misleading but not outright false.

The motivation of this project partially comes from fake news. The paper "Social Media and Fake News in the 2016 Election," a collaboration between Hunt Allcott from New York University and Matthew Gentzkow from Stanford University, investigated the motivations for generating fake news, it's audience and the consumption of fake news, and studied people's perception of both true and false news related to 2016 Election campaign (Allcott and Gentzkowm, 2017).

A primary motivator to create fake news is to profit from monetary profit by attracting huge online click volume. Others do so to promote political ideologies or alternatively attack opposing ideologies. According to Allcott and Gentzkow (2017), consumers have incentives to consume precise and unbiased news for their own good, but they also receive psychological security from news that confirms their prior beliefs. In this vein, fake news often targets the already present-fears and worries of those it targets. According to a survey from Pew Research Center, 64% of U.S. adults expressed a sense of confusion caused by made-up stories regarding basic facts of current events (Barthel, Mitchell, & Holcomb, 2016). In addition, the study indicated that 23% of U.S. adults have shared fake news with friends and others (Barthel et al., 2016).

Potential social costs of fake news include:

- Skew news consumers' beliefs about what is true.
- Undermine the ability of the democratic process to select high-quality candidates.
- Undermine the credibility of news producers, both legitimate and fake ones.
- Reduce demand for high-precision, low-bias reporting (Barthel et al., 2016).

There are numerous recent examples of 'Fake News'. In fact, according to Webb (2017), Facebook has recently announced that ads purchased by Russian forces have reached over 10 Million people in the United States. Examples of Fake News include: "Hillary Clinton endorsed Donald Trump as an honest person who should run for president in 2015", or the "Hillary Clinton campaign had murdered 17 people who tried to leak information" (Allcott and Gentzkowm, 2017). None of these statements are true. They are, however, are rooted in fears and biases many people already had. In large part, reducing the amount of misinformation on social media platforms and the Internet in general begins in the hands of the platforms, not the people. However, no solution has yet been presented. Thus, our project aims to enable people to more quickly parse through a document to determine how factual it is, enabling them to make their own determination of the information they are reading in an informed manner.

1.3.2 Polarization

Lelkes (2016) found that modern society has become increasingly polarized. While this varies depending on the issue, there has been an increase of coupling ideology and identity; there is a strong perception of this which seems to further fuel polarization. Self-identification as a partisan, one who has particular political leaning rather than a set of personal beliefs, has led to growing hostility - “affective polarization” (Lelkes, 2016; Abramowitz and Saunders, 1998; Bartels, 2000). According to Hetherington et. al. (2015), inter-party hostility can lead partisans to distrust the government when the other side is in control, leading to dysfunction.

Opinionated documents are not always balanced. Being able to discern if a document is highly negative, or relatively balanced by at least lacking vitriol may help to encourage intelligent conversation.

1.3.3 Recent developments

Fake news on social media: Given the popularity of social media platforms, fake news can spread very quickly. Besides trustworthy sources, there are sources that do not filter their contents or post made-up stories. This is a slice of the advertising revenue that comes from clicks as people follow the links to their web-pages.

Based on the demand for filtered facts, tech companies are taking steps. Facebook has updated its advertising policies to reduce spam sites and allowed its users to report fake articles (Guynn, 2017; Moon, 2017). Google is also improving its search algorithms to take into account the accuracy of information when displaying search results. Additionally, Google is helping fact-checking organizations such as Full Fact, which is creating an automatic fact-checker that can monitor claims made on TV, newspapers, parliament, or the Internet (Kolodny, 2016). However, the project received negative feedback since clients did not want to be told what was true or not. According to Alexios Mantzarlis, director of the International Fact-Checking

Network, people prefer to be given resources to fact-check information themselves. In fact, he is planning to develop a database of sources that fact-checkers use and make it available to the public. Similarly, our project aims to provide a tool for automatic objective sentences extraction, the very first step of the fact-checking process.

Based on the public needs, this project, Fact or Fiction, attempts to isolate fact from opinions to provide a common ground for people with different ideologies to come and discuss facts that are universally accepted as true. In addition, it provides clients with sources about entities such as links to related articles so they can make the judgment themselves.

Chapter 2. Background & Related Work

Our project focuses on machine learning classification of sentences as subjective vs objective. Classification is the task of determining the category of an object from a set of categories. For example, one might classify images to images of dogs and images of cats, or classify spam versus non-spam emails. Text classification is an active research area in Machine Learning (Aggarwal, & Zhai, 2012), (Kim, 2014). In fact, the subfield of Machine Learning that focuses on human language is called Natural Language Processing (NLP). There are numerous publications aiming to classify sentence as objective or subjective (Yu & Hatzivassiloglou, 2003), and extract opinions from texts (Wilson et. al, 2005). In this chapter, we discuss some of the related work that classifies opinions and facts at either the document-level or, closer to our work, the sentence-level representation.

2.1 Literature Review

2.1.1 Naive Bayes Sentence Classification of Wall Street Journal data

In 2003, Yu et al. introduced methods to discriminate facts and opinions at both document and sentence levels. In order to classify the documents, a Naive Bayes classifier was used on a dataset composed from 4000 Wall Street Journal articles labeled as a) Editorial, b) Letter to editor, or c) Business and d) News. Articles of types Editorial and Letter to editor were mapped to opinions, while articles having types Business and News were mapped to facts. A Naive Bayes classifier is based on Bayes Theorem. Based on class occurrences in training data, the Naive Bayes classifier assigns test data to its mostly likely class that is most likely to be correct, by choosing the class with highest posterior probability.

Evaluation of 4000 other Wall Street Journal showed that this technique achieved 97% F1-score (the harmonic mean of precision and recall). Precision and recall are both measures of accuracy which account directly for the relative false-positive and false-negative ratios. Precision measures the proportion of sentences that are correctly labeled positive.

For sentence classification, three methods have been proposed in the literature based on the expectation that documents classified as opinions tend to have mostly opinion sentences, whereas documents categorized as facts tend to have mostly factual sentences. For the first method, they compared a sentence in question with sentences in opinion and factual documents on a similar topic, then assigned the sentence to the category with the highest average similarity

score across all sentences in that category. The second method involved training a Naive Bayes classifier on sentences within opinion and factual documents. The third method was an improvement of the second one in that it involved cascading multiple Naive Bayes classifiers, thus boosting classification accuracy.

With the help of human evaluation, they determined that while variants of the last two classifiers outperformed the first one, they were only sensitive to opinion sentences and did poorly on factual sentences. For example, recall and precision scores for classifying opinions of a certain variant could be as high as 91% and 86% respectively (F1-score = 88%). However, for classifying facts, they were only 44% and 53% respectively (F1-score = 48%).

To our knowledge, this paper is the first attempt to classifying objective and subjective sentences. Most of the features proposed in this paper were used in our experiment.

2.1.2 OpinionFinder

Another research paper created OpinionFinder (Wilson et. al, 2005), which is a large pipeline aiming to automatically identify when opinions, sentiments, speculations, and other private states are present in a piece of text. This system was implemented using a number of open source libraries including Sundance - a partial parser that provided semantic class tags, Named Entities, patterns that correspond to subjective language; OpenNLP for tokenizing, sentence splitting, and part-of-speech tagging; Abney stemmer for stemming. There were 2 subjectivity classifiers used in the system. The first one, using the Naive Bayes classifier at its core, achieved an accuracy of 76%. The second one was a rule-based classifier and reported to have about 91.7% subjective precision and 30.9% subjective recall (F1-score = 46%).

The goal of this system is similar to our system in that it uses a machine learning model to classify subjective and objective sentences. However, this system focuses more on finding subjective sentences, whereas our system targets for highlighting objective sentences.

2.1.3 Passive Aggressive on news data

Later in 2007, Stepinski et. al (Stepinski & Mittal, 2007) described a fact/opinion sentence classification method based on a Passive-Aggressive algorithm trained on unigram, bigram, and trigram features. An n-gram is an ordering of n-words; unigrams are single words whereas trigrams try to capture the relative positions of words. The authors trained multiple classifiers in such a way that a classifier that came before were used to reduce noise from input data, and the cleaner data was then used to train the subsequence classifier. After training on a set of sentences from 70,000 fact-based articles and 70,000 opinion-based articles, they reported the average F1-score of the cross-validation to be 85%.

2.1.4 Convolutional Neural Network

In 2014, Kim trained a subjective/objective sentence classifier with a Convolutional Neural Network (CNN) on top of word vectors, which were pre-trained on 100 billion words of Google News using an unsupervised model (Google, 2013). CNN is a type of neural network whose architecture is inspired by the organization of the visual cortex in animal brains.

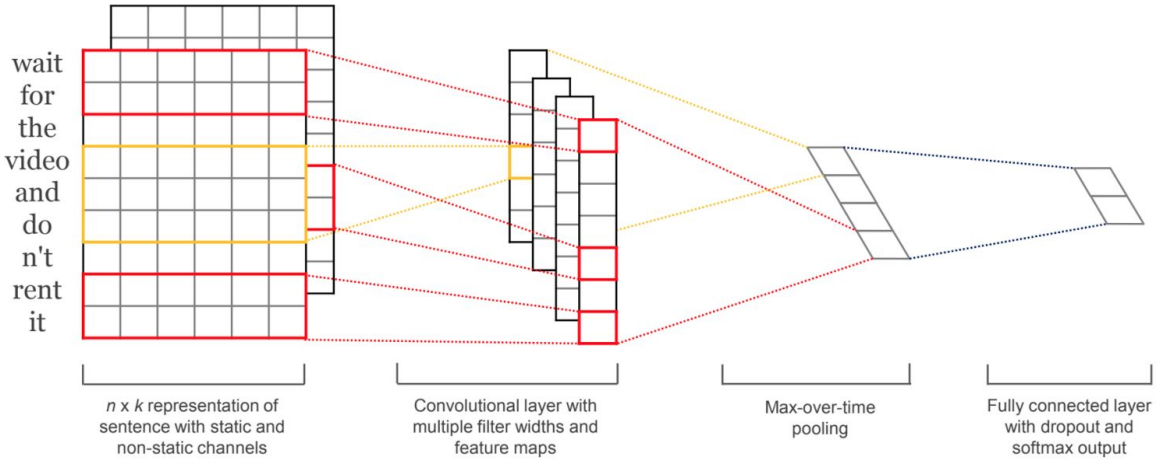

Figure 2. The high-level structure of the CNN model.

It can be seen from the Figure 2 that their model architecture was rather simple: there was a single convolutional layer, one pooling layer, and one fully connected layer. Despite its simplicity, their evaluation experiments proved that such a CNN architecture performed well on a variety of tasks. In the task where the goal was to classify a sentence as being subject or objective, the CNN yielded an accuracy of 93.4% on its best variant. It is worth noting that for the variant where there was no fine-tuning for the word vectors, the neural network already achieved 93.0% accuracy. This result showed that the pre-trained word vectors were quite versatile and could be applied to a vast array of tasks.

As can be seen in the latter sections, this was the model that we used in our final system.

2.1.5 Discussion

By reviewing the proposed methods in a chronological order, we recognize two shifts in how the authors tackled the subjective/objective classification problem. First, there was a shift from classifiers that rely on hard-coded and hand-crafted features to classifiers that can autonomously learn the representation of textual data. Second, the choice of classifiers has also been moved

from being generative (these classifiers try to model a probability distribution of each class; e.g. Naive Bayes) to being discriminative (these classifiers merely find the best boundaries between classes; e.g. Logistic Regression, Neural Network). Each of these shifts resulted in significant improvement in performance.

The initial work in the field by (Yu et al., 2003), (Wilson et. al, 2005), and (Stepinski et. al, 2007) were similar in their choice of Naive Bayes classifier. Besides low accuracy, their other shortcoming was in how their data was obtained. Specifically, they expected the sentences would have the same characteristics (subjective and objective) as their source articles. However, most of the time an article can contain any number of sentences of both types. In general, rigid statistical theories and machine computation were not able to identify nuances in human communication. Although (Stepinski et. al, 2007) selected a potentially better classifier, it also suffered from the lack of a reliable dataset.

2.2 Technical Review

2.2.1 Machine Learning Techniques

Machine learning involves extracting patterns and information from data to create model trained to perform a certain task. For our purposes, the task is sentence classification, training data (sentences) are labeled with a classification (e.g. “objective” or “subjective”), and the task is measured by a loss function (a measure of how well the model fits training data). The model attempts to classify the training data, and the loss function gives a measurement of how well the model did relative to the previous iteration. Then the model adjusts its parameters in order to minimize the value from the loss function. The model improves upon itself through many of such iterations. There are different training routines; some models can be solved analytically by taking the derivative of the loss function with respect to the model parameters, whereas others are trained iteratively with methods such as gradient descent (Burges et al., 2005).

Shallow Learning

There are numerous traditional machine learning models (or so-called shallow models), which have typically been trained on features selected by researchers and practitioners. First, to handle textual data, it must be transformed into a numerical and vectorized format. Shallow learners, (e.g. Support Vector Machines (SVMs) and Naive Bayes), generally use Bag of Words representations. Each item of text (the document) is transformed into a vector where each index represents the count of a corresponding word, with the length of the model being the number of unique words. Beyond efficiency concerns, which are largely ameliorated by optimization techniques such as hashing, this data is extremely sparse and does not include semantic meaning. So, shallow learners often additionally rely upon features such as n-grams, tf-idf, as well as

part-of-speech annotations. N-grams are extended Bag of Words, which are built on vocabularies of n words that appear contiguously in a document (Frakes, 1992).

Naive Bayes (NB) is a classification algorithm that works by choosing the class that is most likely given a learned distribution of class occurrences in the input/training data. The model is based on the Bayes Theorem, which states that the probability that a class *y* can be assigned to a given input example $(x_1, x_2, \ldots, x_n)$ is the product of the posterior probability of the input example given the class and the prior probability of that class (Eq. 1)

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)} \qquad (1)$$

The meaning of these terms are what their names suggest: the posterior probability is the probability of the data given the class, and the prior probability is the probability of any data being of that class. Note that this expression is normalized by the likelihood of the data itself, but this value is equal for each class, and can thus be ignored. The "naive" in its name comes from the assumption that all of the values x_i are independent of each other, therefore

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y), \qquad (2)$$

As a result, we train the model by estimating $P(y)$ and $P(x_i \mid y)$ from the training data.

Support Vector Machines (SVM) is a classification algorithm that attempts to fit a hyperplane (boundary) between features extracted from two classes. To do this, it uses a hinge loss that incorporates a flexible margin making it possible to capture complex patterns. It can also incorporate more complex structures by using kernel functions (Friedman, 2001). As shown in Figure 3, SVM attempts to find a boundary that maximizes the margins between the two classes. The examples on the margins are called the support vectors.

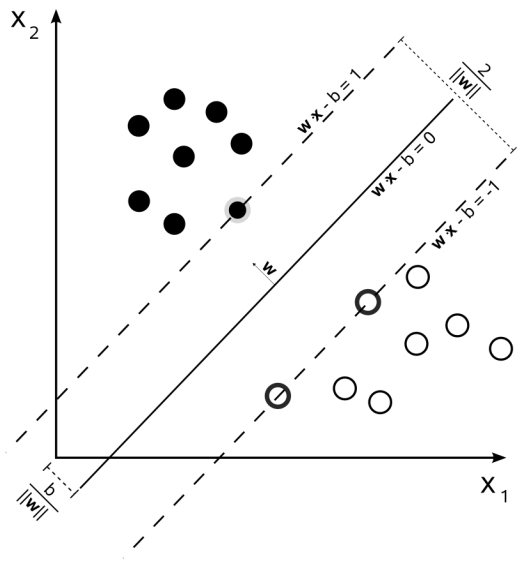

Figure 3. Illustration for the SVM classifier (source: Wikipedia).

Deep Learning

Neural networks are versatile and increasingly ubiquitous models that have achieved impressive performance in nearly every machine learning tasks, and introduced new capabilities previously impossible, especially on large datasets. These models at their core are stacked layers of simple perceptron models trained together with backpropagation. There are numerous strategies, techniques, and optimizations that increase the stability and accuracy of these models[1]. Additionally, Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNNs), two popular general architectures, both have had marked success in many applications, including NLP (Fernández et. al, 2007; Kim, 2014; Krizhevsky et. al, 2012; Mou et. al, 2016).

An Artificial Neural Network (ANN) is a computational model based on the structure and functions of biological neural networks (Artificial Neural Network (ANN), n.d.). The neural network changes, or learns, based on the flow of information. In addition, RNNs are a type of advanced ANN that involves a directed cycle in memory. RNNs build on earlier types of networks with fixed-size input and output vectors. RNNs have had success with text generation, classification, and time series forecasting (Gers, 1999). CNNs instead involve local operations on sections of the input to extract features. Hierarchical layers in CNNs are able to extract high-level features from lower level features extremely effectively, and in fact there are numerous examples where these self-learned features can be interpreted by humans (Olah, 2017). CNNs have traditionally been used for computer vision tasks like image classification and object detection, but also have been successful in some NLP tasks like text classification (Wang, 2012).

[1] We elide descriptions and refer the reader to http://www.deeplearningbook.org/ (Goodfellow, 2014).

2.2.2 Microsoft Technologies

Microsoft provides many technologies which cover various aspects of our project. In this section, we review each of these tools and discuss how we use them in our project.

ASP.NET Core

ASP.NET Core is an open-source software framework developed by Microsoft for building web applications and services on Windows, Mac OS, and Linux. The prototype was written using ASP.NET, but we decide that ASP.NET Core is a much better option for the project. According to the documentation provided by Microsoft (2016), ASP.NET Core should be used when there are cross-platform needs, or when the systems need to have high performance and be scalable. On the other hand, ASP.NET is used when the app currently uses it, when the app uses third-party libraries or NuGet packages that are not available in ASP.NET Core, or when the app uses a platform that does not support. However, none of those reasons for using ASP.NET apply in our project.

Language Understanding Intelligent Services (LUIS)

LUIS is a set of web services for Natural Language Understanding (NLU). The primary purpose of LUIS is to provide a service that allows others to build applications. Given a phrase, LUIS will parse the phrase into entities, and then give confidence scores on the actions and purpose of the phrase. LUIS can also be used to get sentiment scores on entities, provide a framework to build chatbots upon, and is multilingual. Any client application that converses with users, such as a dialog system or a chatbot, can pass user input to a LUIS app and receive results that provide natural language understanding.

However, its utility reaches beyond a powerful NLU capabilities; it provides a rich API which allows developers to gain fine-tuned control over their models. It operates around three key concepts: utterance, intent, and entities. An utterance is textual input, and this utterance is then parsed into its intents, the verbs in the phrase, whereas the entities are the nouns. We can leverage these capabilities as we construct our service to differentiate between facts and opinions.

An example use case of LUIS: a booking agency is creating a website chatbot to help decrease the time it takes customers to buy tickets. One of the customers enters, “ Book me a flight to Cairo”. LUIS can then be used to determine the *intent* of the request to be “BookFlight” as well as determine the location to be “Cairo”. From here, the agency could direct the user to upcoming flights to Cairo from the user’s location. Parsing and understanding unstructured text is not easy,

and with this capability, LUIS allows this booking agency to build a consistent, and trustworthy chatbot.

The process for creating an application with LUIS is very similar to the traditional machine learning paradigm. First, we define, or restrict, the intents, utterances, and entities to those the application expects to handle. Then, we construct features, train, and test the model. An advanced feature of LUIS is that it allows for Active Learning, wherein it informs the developers of utterances that it had a difficult time scoring. This allows LUIS to update the model while it is being used, and continually improve.

We used LUIS in the Fact or Fiction prototype due to time constraint. A high-level diagram of how LUIS fits into this project is presented in Figure 7 in section 3.2, where the role of the ML Service can be fulfilled by LUIS or any other Machine Learning Service with different models.

Bing Entity Linking Service

An entity is a *thing* that exists independently, as a subject or object. It could be physically existing or just an abstraction. Entities in languages don't serve syntactic purposes. When processing natural languages, a word can be used as multiple entities. For example, "times" is a named entity, it may refer to two separately distinguishable entities such as "The New York Times" or "Times Square". The Microsoft Entity Linking Intelligence Service recognizes and identifies each separate entity within a paragraph or text, based on the context it is given. An example input and output is now presented for illustration purposes.

Example input to the Microsoft Entity Linking Service:

> Worcester Polytechnic Institute has a Major Qualifying Project. Four students are going to Microsoft, Cambridge to complete this project. This project utilizes Bing Entity Service to highlight entities.

Results of input:

> **Worcester Polytechnic Institute** has a **Major Qualifying Project**. Four students are going to **Microsoft**, Cambridge to complete this project. This project utilizes **Bing Entity Service** to highlight entities.

Entities such as "Worcester Polytechnic Institute", "Major Qualifying Project", "Microsoft" have been identified in the text and highlighted.

Microsoft Cognitive Toolkit (CNTK)

The Microsoft Cognitive Toolkit (CNTK) is a unified deep-learning toolkit. It supports 64-bit Linux and 64-bit Windows. CTNK can be included as a library in C++, C# or Python programs.

It can also be used as a standalone tool by using the model description language (Brainscript). CNTK allows to easily realize and combine popular model types such as feed-forward DNNs, Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs/LSTMs). It implements learning time speedup optimization algorithms such as Stochastic Gradient Descent (SGD), as well as error backpropagation, learning with automatic differentiation and parallelization across multiple GPUs and servers (Microsoft Cognitive Toolkit Github, n.d.).

CNTK have highly optimized, built-in machine learning models and techniques that are compatible Python, C++, or BrainScript. It makes resource usage efficient by sharing memory among multiple GPUs/machines. CNTK also provides access to high-speed resources such as Azure GPU and Azure cloud storage networks (Microsoft Cognitive Toolkit Features, n.d.). In this project, CNTK is used to create deep learning models, particularly RNN and CNN.

Azure Cloud Computing Platform & Services

Azure is a Cloud Computing Platform developed by Microsoft with 42 regions as shown below, which is the largest number among cloud service providers.

Figure 4. Microsoft Azure Region Map (source: buildazure.com).

Azure provides support for both Linux-based and Windows-based operating systems, major mobile platforms, and a diverse range of programming languages, databases, frameworks, and devices. In addition, Azure is a consistent hybrid cloud platform. It allows clients to distribute and analyze data seamlessly across cloud and on-premises. The hybrid consistency is also present in application development, identity management, management and security, and data platform. Additionally, users can build and deploy custom AI models at scale, or utilize Microsoft's AI open source data and resources. Lastly, Azure is trusted by 90% of Fortune 500 companies. It is also the most trusted cloud for U.S. government institutions (Microsoft Azure, n.d.). Azure Security Center allows users to detect and mitigate threats with a global view of

Azure resources. All the services part of our application are hosted on Azure, including the back-end, the Machine Learning services, and the external knowledge-based service.

2.2.3 Other Open Source Technologies

React and Redux

React is a Javascript library that allows developers to build dynamic web applications that can change data without reloading the page (React, 2017). React is used for user interfaces, corresponding to the view in the Model-View-Controller (MVC). React is highly versatile and does not prescribe a framework. Interfaces built in React rely heavily on the composition of Javascript XML (JSX), which are then rendered by React to the DOM. React maintains an internal representation of the tree-structure of the JSX elements, allowing it quickly re-render only what is necessary.

Redux is a Javascript library designed to manage application state and data flow. It is often used with React or Angular to build user interfaces. It uses reducers and actions, loosely corresponding to the model and controller in MVC pattern, respectively. Actions represent the events or "what happened" and reducers update the state according to these actions (Redux, 2017). In order to prevent cyclic dependencies and keep the application predictable, redux uses an immutable state which is copied and modified. This immutability prevents unexpected behavior and improves developer tools.

React and Redux were used to program the front-end of the Fact or Fiction application. When developing a highly dynamic front-end, writing raw Javascript and manipulating the DOM trees are notoriously tedious and error-prone. Therefore, this library combination significantly streamlined the design process and boosted development speed.

InferSent

Conneau, Kiela, Schwenk, Barrault and Bordes (2017) from Facebook AI Research group developed InferSent, which is a method to represent a sentence using an n-dimensional vector by performing supervised learning on the Stanford Natural Language Inference Datasets. The implementation of their work has been published as an open source project under the name InferSent. Using the InferSent Python module, we can quickly embed an arbitrary sentence into a 4096-dimensional vector space. Two sentences that are semantically similar to each other will generally have their corresponding InferSent vectors to be close to each other in vector space. We utilize this property to display sentences that are related to a given sentence.

Chapter 3. Methodology

In this chapter, we explain our methodology for solving the problems stated previously and reaching our goal of sentence classification. This chapter has three main sections. In section 3.1, we describe how we collect and pre-process the data and experiment on different classification models. In section 3.2, we outline the requirements for the final application and our design to satisfy these requirements. In section 3.3, we present our interviewing and surveying methods to collect feedback from real users in order to assess the usability of the Fact or Fiction application.

3.1 Machine Learning

Figure 5 illustrates the steps that we took to produce the final machine learning model. These steps are: gathering the data, preprocessing the data, constructing shallow models and deep models, and finally training and testing these models. For shallow learners, we extracted features from the text and then transformed the data before feeding them into the models. For deep learners, we did not have to hand-pick features. Using the same sets of data, we trained all of the models we constructed, including Naive Bayes, SVM, RNN and CNN, and recorded results for each model.

3.1.1 Data Collection and Labeling Rules

Data Collection

In the early stage of our project, we attempted to collect dataset in order to train machine learning models. We found Cornell's Movie Review Data (Pang & Lee, 2004) that contained sentences labeled as objective and subjective. This source contains two sets of data: 5000 objective sentences extracted from movie plot descriptions and 5000 subjective sentences extracted from movie reviews.

Since our envisioned use case targets news articles, we needed to collect more labeled text corpus with a wide range of topics and political biases. As a result, we gathered text corpus from news sources including BBC (Greene & Cunningham, 2006), CNN (Qian & Zhai, 2014), Fox News (Qian & Zhai, 2014), and BuzzFeed (BuzzFeedNews Github), and manually labeled 5000 sentences ourselves. In order to create a consistent dataset, we researched the literature to understand the distinguishing factors between objective and other statements, and created a set of rules that we strictly followed in the process of labeling the data. These rules are listed in detail in the next section.

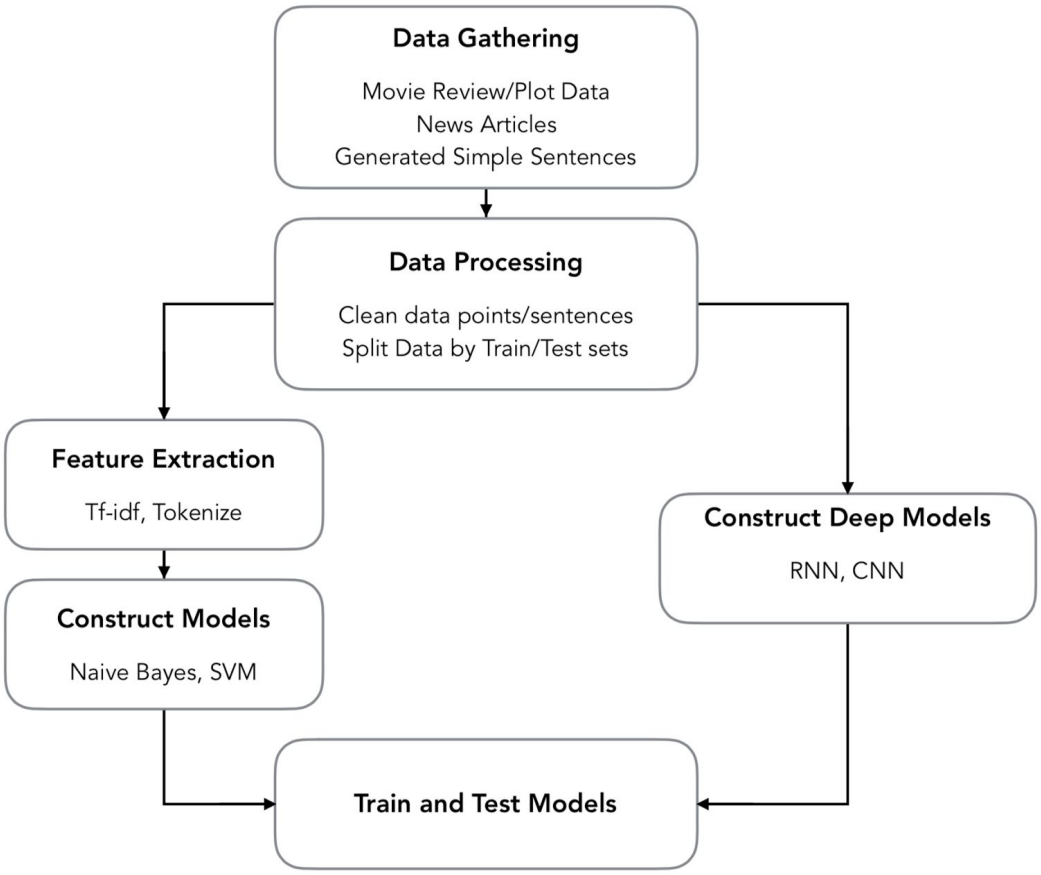

Figure 5. Model training pipeline.

After completing with hand-labeling the news dataset, we found a slight disparity in subjective and objective sentences, as the dataset had more subjective sentences. We then discovered Wikicorpus (Reese, Boleda, Cuadros, & Rigau, 2010), which is a collection of articles from Wikipedia, and randomly picked 500 objective sentences and 20 subjective sentences to merge into the dataset. Some example sentences from this corpus are shown in Table 1.

Table 1. Example sentences from the Wikicorpus.

Objective sentences	**Subjective sentences**
- Ethnic studies is an academic discipline dedicated to the study of ethnicity. - Macke was born in Meschede, Germany. - Boeing retired from the aircraft industry in 1934. - For the past 150 years, lighting technology	- In many rural shrines, she has no form - just a granite stone with a sharp tip, almost like a spear head. - Kent is sometimes known as the Garden of England because of its agricultural influence, extensive orchards and hop-gardens. - Probably the most significant geographical

was mainly limited to incandescence and fluorescence.	feature of Kent is the White cliffs of Dover. - Like many outstanding artists of her time, Bass experienced a revival of interest.

In addition, in order to better recognize a more informal writing style than news articles, we generated 1500 objective sentences and 1500 subjective sentences by substituting appropriate words into simple templates. Table 2 shows examples of these sentences.

Table 2. Example sentences generated from templates.

Templates	**Sentences**
<noun> is <adjective> (objective)	The sky is blue. The Earth is round. This car is red. An orange is perishable.
This <noun> is too <adjective> (subjective)	This person is too smart. This dog is too good. This job is too awful. This building is too high.

In total, we had 8116 objective sentences, and 8631 subjective sentences, yielding a total of 16747 data points.

After aggregating all sentences in a single comma-separated file, we started preprocessing each of them to conform to the following criteria:

- All punctuations are eliminated.
- All tokens are separated by a single space.
- URLs, such as www.google.com, are replaced with <URL>
- Numbers are replaced with <NUM>
- Time, such as 12:00 am, are replaced with <TIME>

We performed the last 3 substitutions to reduce the variability in our dataset and the size of the dictionary. Note that this preprocessing is also applied to the new data coming into the final deployed system.

Objectivity/Subjectivity Annotation Rules

Definition

We found a definition from Wiebe, Bruce, and O'Hara's *Development and use of a gold-standard dataset for subjectivity classifications* (1999): "If the primary intention of a sentence is an objective presentation of material that is factual to the reporter, the sentence is ***objective***. Otherwise, the sentence is ***subjective***".

Example (Wiebe, Bruce, & O'Hara, 1999):

- At several different levels, it's a fascinating tale. *Subjective sentence.*
- Bell Industries Inc. increased its quarterly to 10 cents from seven cents a share. *Objective sentence*.
- Northwest Airlines settled the remaining lawsuits filed on behalf of 156 people killed in a 1987 crash, but claims against the jetliner's maker axe being pursued, a federal judge said. *Objective speech-event sentence*.

Private state is a general term that covers mental and emotional states, which cannot be directly observed or verified (Greenbaum, Quirk, Leech, & Svartvik, 1985). For example, we can observe evidence of someone else being happy, but we cannot directly observe their happiness. In natural language, opinions, emotions, and other private states are expressed using subjective language.

Rules

A sentence is **subjective** if:

1. It contains adjectives or adverbs that have an orientation (e.g. beautiful, ugly, simple, good, bad) as opposed to adjectives that do not have an orientation (e.g. domestic, medical, red) (Hatzivassiloglou & Wiebe, 2000).
2. It contains explicit private state (e.g. think, believe, hope, want) or a private state mixed with speech (e.g. berate, object, praise) (Wilson & Wiebe, 2003).
3. It contains events that are not significant, speculative or not real (called "minor private states and minor speech events" in Wilson and Wiebe's study). Examples (Wilson & Wiebe, 2003):
    - ***Such wishful thinking*** *risks making the US an accomplice in the destruction of human rights.* (not significant)
    - *If the Europeans* ***wish*** *to influence Israel in the political arena...* (in a conditional, so not real)

- *And we are seeking **a declaration** that the British government **demands** that Abbasi should not face trial in a military tribunal with the death penalty."* (not real, i.e., the declaration of the demand is just being sought)
- *The official **did not say** how many prisoners were on the flight.* (not real because *No one who has ever studied realist political science **will find this surprising**.* the saying event did not occur)
- (not real since a specific "surprise" state is not referred to; note that the subject noun phrase is attributive rather than referential (Donnellan, 1966))

4. If a sentence contains a quotation comprised of multiple sentences, look into the sentences in the quotation separately.
   - "Today is a good day. The sky is blue", said Bill Gates. (**2 sentences**, the first one is **subjective**, the second one is **objective**).
5. For sentence contains nested source (A said that B thought that C did something was bad) (Wilson & Wiebe, 2003), we consider the writer's point of view. For example:
   - *The South African Broadcasting Corp. said the song "Freedom Now" was "undesirable for broadcasting."*. According to [4]: "there is no uncertainty or evaluation expressed toward the speaking event. Thus, from one point of view, one might have considered this sentence to be *objective*. However, the object of the sentence is not presented as material that is factual to the reporter, so the sentence is classified as **subjective** (or subjective speech-event sentence to be exact)".
   - *Bill Gates said he recently started using an Android smartphone*[2]. By the same logic, this sentence is **objective**.
6. If a sentence contains modal verbs: can, must, should, etc:
   - *You should not leave the light on when you go to sleep.* **subjective**[3]
7. Conjured causation relationships:
   - *Thanks to Donald Trump, 200 Marines were able to see their families that day.* **subjective**

The rest is **objective** if they are **NOT**:

- Incomplete sentence. E.g. *Hey!*
- Questions. E.g. *Who are you?*
- Imperative sentences. E.g. *Go to sleep, Charlie.*

---

[2] "Even Bill Gates uses an Android phone now - CNBC.com." 26 Sep. 2017, https://www.cnbc.com/2017/09/26/bill-gates-switches-to-android.html. Accessed 1 Nov. 2017.

[3] "Modality: meanings and uses - English Grammar Today - Cambridge ...." http://dictionary.cambridge.org/grammar/british-grammar/modality-meanings-and-uses. Accessed 1 Nov. 2017.

3.1.2 Classification Models

We experimented with a number of machine/deep learning classification models, including both shallow and deep learners.). We focused on comparing prior work and building from there, notably Naive Bayes classifiers and Convolutional Neural Networks (CNN) (Krizhevsky, Sutskever, & Hinton, 2012). Besides Naive Bayes and CNN, we also experimented with Support Vector Machines (SVM) and Recurrent Neural Network (RNN) (Mikolov, Karafiát, Burget, Cernocký, & Khudanpur, 2010) in combination with Long Short Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997).

We partitioned the dataset into train and test sets. In order to avoid data snooping, we further divided the training dataset into a validation set which we used to inform our model design as we experimented. Although we were generally interested in achieving good classification accuracy, we also measured the recall, precision, and F1-score to ensure that we created a balanced and generalized model. After initial experimentation and validation, we tested our final models on the test sets and moved forward using the best performing model. As we built these classification models, we continuously integrated our classification model back into our core server to allow easy deployment.

3.1.3 Feature Extraction and Model Training

As our training data is in the form of text, we need to extract numerical features from our data for the SVM and Naive Bayes models to be able to consume. Our features included:

- *Tf-idf of unigram, bigram, and trigram.* Term Frequency-Inverse Document Frequency, or tf-idf, measures the relative importance of different words that appear in the document by normalizing the term frequency (tf) with the inverse document frequency (idf); this weights common words as less important and rare words as more important (Frakes, 1992). Transforming text data into this feature is a popular method for vectorizing text. This feature was used in both (Yu et. al, 2003) and (Stepinski et. al, 2007).
- *Part of speech tag count.* For example, "heat water in a large vessel" would be tagged as "<verb> <noun> <preposition> <determiner> <adjective> <noun>". Wiebe et. al (1999) pointed out that the presence or absence of a particular part of speech can be a good signal for determining subjectivity.
- *Word sentiment score.* This feature was included in the classifiers in (Yu et. al, 2003). Each word is assigned with one of the three sentiment scores: positivity, negativity, and objectivity. We obtained such assignments from the SentiWordNet lexical resource[4] (Baccianella, Esuli, & Sebastiani, 2010).

[4] http://sentiwordnet.isti.cnr.it/

The result of the feature extraction process was a sparse matrix where each row corresponded to a sentence and each column corresponded to a feature, e.g. a tf-idf value of a unigram, count of a part of speech.

In the Naive Bayes classifier, we used the multinomial distribution to model our data. For the SVM classifier, we experimented with both linear and Radial Basis Function (RBF) kernels.

For deep learning models, we did not manually select and extract features from our text data. This is indeed the beauty of these models because they can automatically learn a new representation of the data. Initially, each word in a sentence was transformed into a one-hot vector. A one-hot vector is a vector whose dimension equal to the size of the dictionary and has a value of 1 at the location corresponding to the word it represents and 0 everywhere else. This one-hot vector was then fed into an embedding layer, which attempted to convert this sparse one-hot vector into a dense vector based on the semantics of the word it represented. One of the most important characteristics of these dense vectors is that when two words are semantically similar to each other, their corresponding embeddings are also close to each other in the vector space. For example, in the t-SNE plot (Maaten & Hinton, 2008) of word embedding shown in Figure 6, we can see that words like "boat", "cruise", and "ship" are in close proximity; whereas "coast", "sea", and "island" are closer to each other.

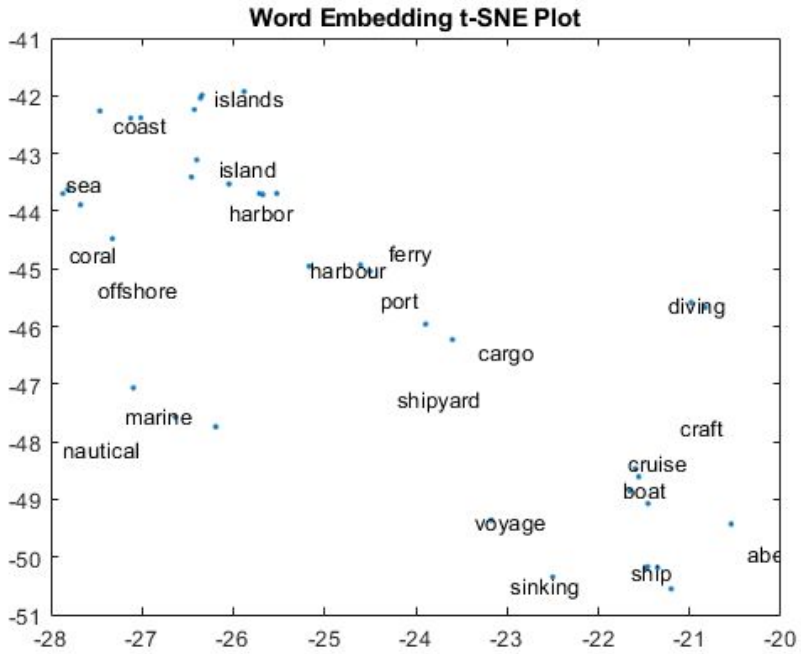

Figure 6. Word embedding visualization. (source: mathworks.com)

We also experimented with both training the embedding layer from our data and loaded it with a model pre-trained on roughly 100 billion words in a Google News dataset (Google, 2013). These

dense vectors were next fed into a deep learning model, which could be an LSTM or a CNN in our case.

Our LSTM model consisted of a recurrence layer and a dense layer. To accelerate the training process, we used the ADADELTA (Zeiler, 2012) training algorithm. The function we optimized was the cross-entropy loss. The cross-entropy loss is a standard loss function for optimizing models for classification. For a given observation the loss is the log probability that the observation labeled correctly.

$$-\sum_{c=1}^{C} y_{o,c}\, log(\widehat{y}_{o,c})$$

C is the number of different class, $y_{o,c}$ is a binary indicator label that is 1.0 for the correct class of the observation o and 0.0 otherwise, and $\widehat{y}_{o,c}$ is the predicted probability for the class c for the given observation o. Note that when entirely correct, $\widehat{y}_{o,c}$ is 1.0 is the predicted probability of the correct class for an observation, the loss is 0. Finally, the expression is negated because logarithms of fractions are negative and a loss function increases with error.

We created a CNN architecture similar to that described in (Kim, 2014). It has an embedding layer followed by convolution, a max pool, and dropout layers. For non-linear activation function, we used the Rectified Linear Unit (ReLU) (Krizhevsky, 2012). ReLU is a simple max function that activates at 0:

$$ReLU(x) = max(0, x)$$

Like the LSTM, the loss function we optimized was the cross-entropy loss. The curve for this function is a line with a slope of 1 cutoff at at 0.

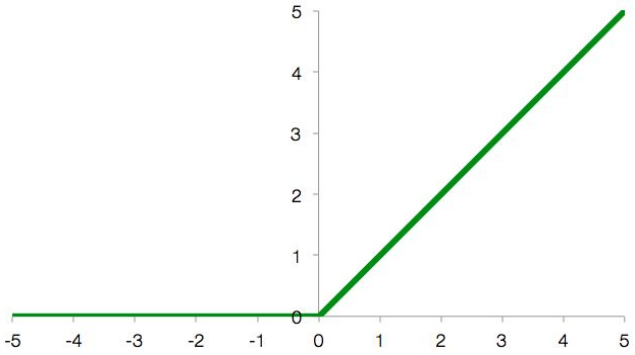

Figure 7. ReLU Curve.

3.1.4 Model Evaluation

To evaluate the effectiveness of our machine learning model as well as the quality of our service, we employ the following metrics.

Accuracy: Accuracy is defined to be the rate at which a classifier correctly classifies examples in the test set to their true categories. We aim to achieve over 80% accuracy for both subjective/objective classifier and opinion sentiment classifier. This direct metric is often biased due to class imbalances but is the most interpretable.

Precision and Recall: Before defining precision and recall, we first define the following terms:

- True Positive (TP) is the number of positive sentences correctly classified as positive.
- True Negative (TN) is the number of negative sentences correctly classified as negative.
- False Positive (FP) is the number of positive sentences incorrectly classified as negative.
- False Negative (FN) is the number of negative sentences incorrectly classified as positive.

Here, "positive sentence" means "objective sentence" in the context of the objective/subjective classifier. We can now define precision as the fraction of sentences correctly classified as objective among all sentences classified as objective by the classifier

$$precision = \frac{TP}{TP + FP}$$

Recall is defined as the fraction of objective sentences returned by the classifier among all objective sentences in the test set.

$$recall = \frac{TP}{TP + FN}$$

The **AUC** is the area under the curve that rate false positive rate. The true positive is the number of true positives normalized by the number of total positives, whereas the false positive rate measures the data points incorrectly labeled as positive, over all the points that were labeled positive. It can be interpreted the probability that a classifier will classify a random positive data point higher than a random negative data point, and is a more holistic measure than accuracy.

While precision and recall are excellent measures to optimize, it is useful to have a single unified metric. Hence, instead of setting a goal separately for precision and recall, we use the **F1-score** metrics, which combines them. The F1-score is defined as:

$$F = 2 \times \frac{precision \times recall}{precision + recall}$$

Our goal is to obtain over 80% on this metrics over the test set for objective/subjective classifier.

3.2 Fact or Fiction Application Architecture

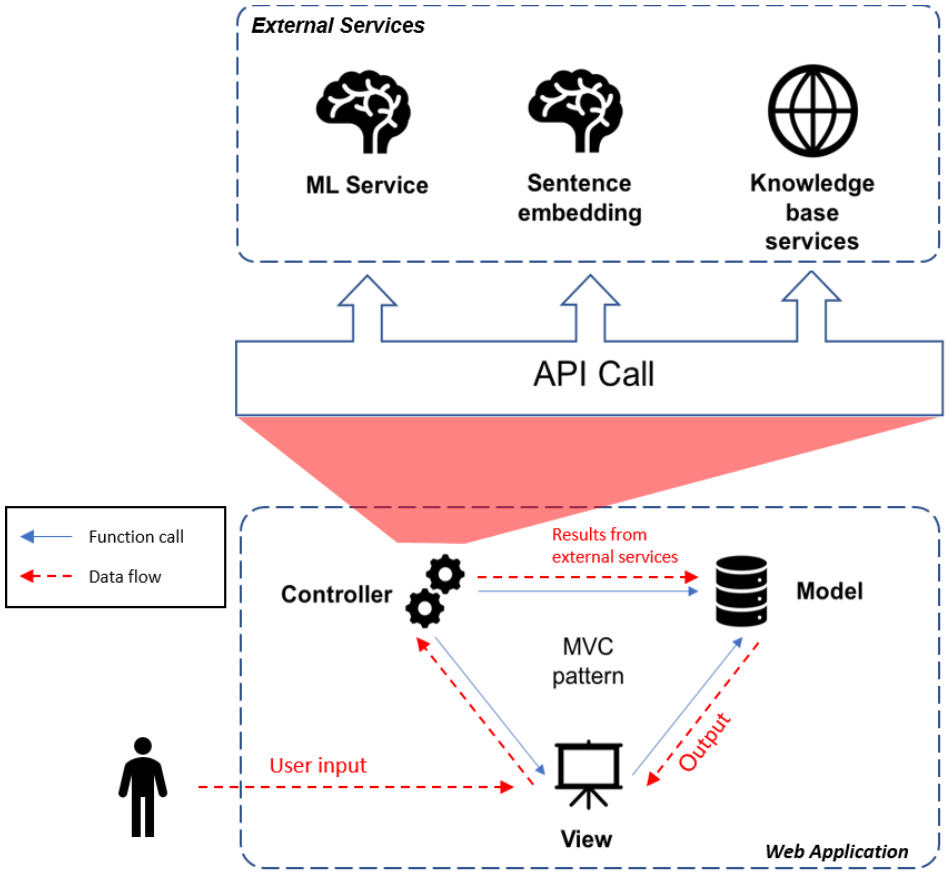

Figure 8. Architectural design of the application.

Figure 7 illustrates the high-level architecture of the Fact or Fiction application. We implemented our application using the Model View Controller pattern. This pattern separates the whole application into three logical groups of components: models, views, and controllers. Models directly manage the application data such as user information and submitted text entries. Views are responsible for displaying data to the end users. The controllers contain the main logic of the application which might involve accessing or updating the models and changing the state of the views. These relationships are illustrated by the blue arrows in Figure 7. By decoupling these three types of components, the MVC pattern significantly promotes low coupling and high cohesion, leading to better reusability and maintainability of code.

In addition to the core application, which includes user authentication and authorization logic, HTTP endpoints logic, and database management logic, we designed the machine learning model service (ML service), sentence embedding service and knowledge base for information related to the sentences as three external entities. The core application communicates with these entities via

an API call, e.g. HTTP request/response. With this design, we were able to easily build the three entities, switch any technologies when necessary and reuse existing third-party services as long as they provided a universal API call for the core application. The interface of these external services should be rather simple: they should all accept a piece of text or a list of sentences. For the ML service, it should return a classification decision for each of the sentences, i.e. objective or subjective. For the sentence embedding, it outputs an embedding vector for each sentence it receives. For the knowledge base services, they ideally consider the semantics of the whole text entry, instead of individual sentences, and fetch related information such as links to relevant articles.

An end-to-end data flow starts from the user entering text input into a view component. This text input is then transferred to a controller, which in turn converts the user input into appropriate formats that conform to each external service and simultaneously sends to all of them. After receiving answers from the external services, the controller transforms these results into the corresponding models, which are later displayed to the user via the view component. This flow is summarized by the red arrows in Figure 7.

3.3 User Experience

3.3.1 Survey

After researching different user experience measures, we decided to use the Standardized User Experience Percentile Rank Questionnaire (SUPR-Q) as it is effective for evaluating websites (Sauro, 2015). The SUPR-Q is an eight-item questionnaire to measure a website's quality and it evaluates the following four factors:

1. *Usability*: how easy it is to use the website.
2. *Credibility*: how trustworthy and dependable the website appears to the user.
3. *Loyalty*: how likely user would recommend and return to the website.
4. *Appearance*: how attractive and clear the website looks (Sauro, 2015).

In order to interpret and evaluate users' responses, we asked each participant to rate the eight questions shown in Table 3 regarding the four factors of the website we built on a scale of one to five (strongly disagree to strongly agree). In addition to the quantitative scores users gave in the SUPR-Q questionnaire, we also obtained qualitative feedback by asking what the users liked and did not like about the website and why.

Table 3. User experience interview questions

Factors	Questions	Score (Scale 1-5) 1 = Strongly disagree 5 = Strongly agree
Usability	The website is easy to use.	
	It is easy to navigate within the website.	
Credibility	The information on the website is credible	
	The information on the website is trustworthy.	
Loyalty	How likely are you to recommend this website to a friend or colleague?	
	I will likely return to the website in the future.	
Appearance	I find the website to be attractive.	
	The website has a clean and simple presentation.	

3.3.2 Interview

We conducted interviews with 5 Microsoft employees to gauge user experience in person. Among these 5 people, four of them are male and one is female. All These were open-ended and unstructured interviews. We would work with a user as they navigated and examined the Fact or Fiction website, noting any comments or feedback. We were specifically looking for usability concerns, trying to make sure that we could rectify any points of confusion.

Chapter 4. Implementation

In order to complete this project within the allotted time frame of 7 weeks, we organized the project into the aspects depicted in Figure 8. The team followed Agile Software Development Life Cycle (SDLC) while developing Fact or Fiction. Agile SDLC enabled iterative and incremental progress and fitted into the seven-week project time frame. We used Visual Studio Online as a team organization tool, specifically to visualize tasks, assign team members to the tasks, and keep track of the work progress. As illustrated in Figure 8, after acquainting ourselves with the work environment, e.g. receiving corporate access, we first designed our initial machine learning models We then parallelized our efforts to simultaneously build the front-end, set up the back-end database servers and machine learning services. After implementing and testing the completed functionality of the platform, we tuned the models and ran user studies to evaluate the user's experience while using the Fact or Fiction website.

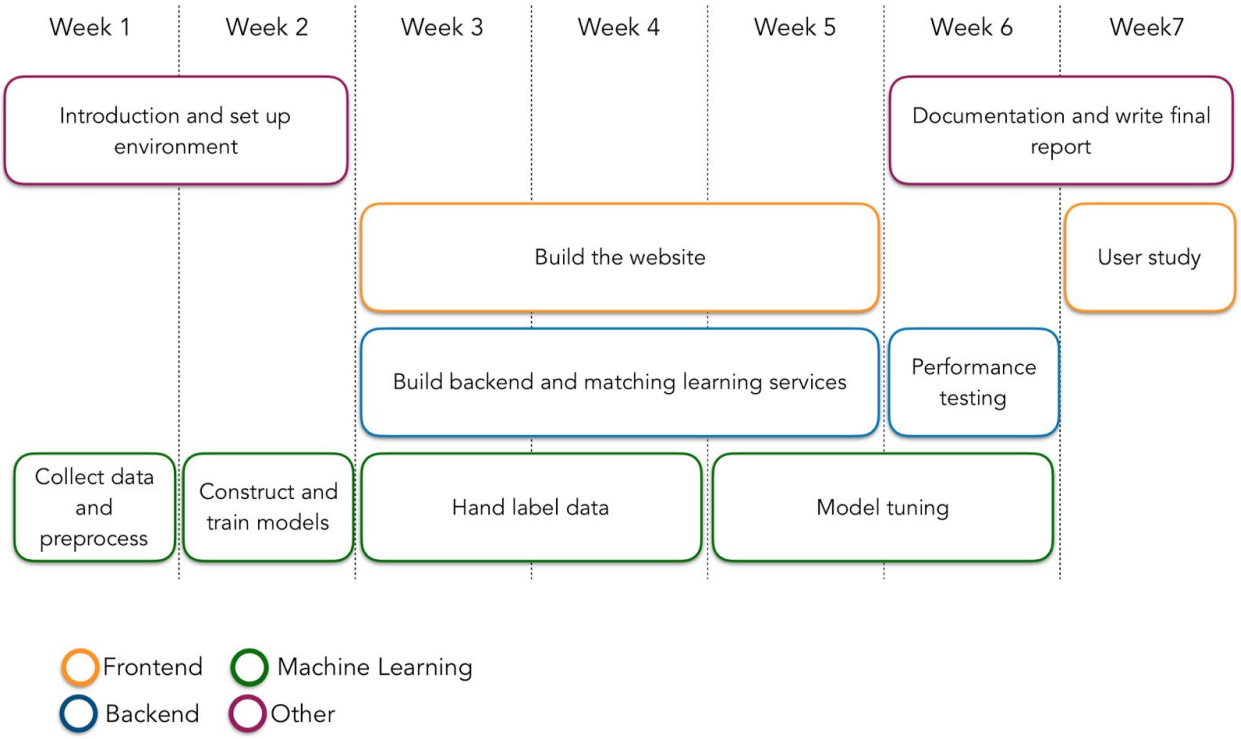

Figure 9. Project timeline.

4.1 Fact or Fiction Application Required Functionality

Based on our objectives, we laid down the following basic requirements that the final application needed to meet.

- The application can classify objective and subjective sentences in text input in real time.
- The user must login in to make a text entry submission.
- The user may log in using an internal account or a social networking account (Microsoft, Facebook, and Google).
- The user can see his/her past text entry submission.
- The user can see other users' submissions on the front page.
- The user can see related sentences by selecting any sentence.
- The application will provide information related to a sentence in order to assist the user in fact-checking.
- The user can vote on the truthfulness of any sentence.
- A user with admin account can view or remove any sentence submitted by any user.

4.2 Back-end and Machine Learning Services

Our first task was to define and build the structure of the service. This requires integration with several different technologies such as ASP.NET Core MVC, Microsoft Entity Linking Service and so on. By designing a simple, but modular, system each module can be worked on in parallel by various team members, and updated individually.

4.2.1 Architecture Overview

Figure 9 shows a high-level overview of the application architecture and the technologies that we used for each component.

We employed the ASP.NET Core MVC framework as the backbone of our product. This framework provides a cohesive and consistent means of communication between the model, view and controller components and a set of convenient libraries for each of them. For the models, we utilized the Microsoft SQL Server to store the data. We also used the Entity Framework Core library as an abstraction layer that hides away all of the nitty-gritty details of SQL commands. For the views, ASP.NET Core MVC provides a template engine called Razor to easily generate HTML pages from the models. In addition to that, we used the React and Redux Javascript libraries to create interactive web pages. For the controllers, besides writing our custom logic to make the application work, we offloaded the implementation of the authentication and authorization feature to the Identity Framework that came with ASP.NET.

To further make our application scalable and easy to modify, we served our machine learning models, including the classification model and the sentence embedding model with InferSent, as separate web services. These custom web services are further described in section 4.2.4. We also obtained references for the sentences from the Microsoft Entity Linking Service and Microsoft Bing Web Search. The application interfaced with all of these services via HTTP API calls.

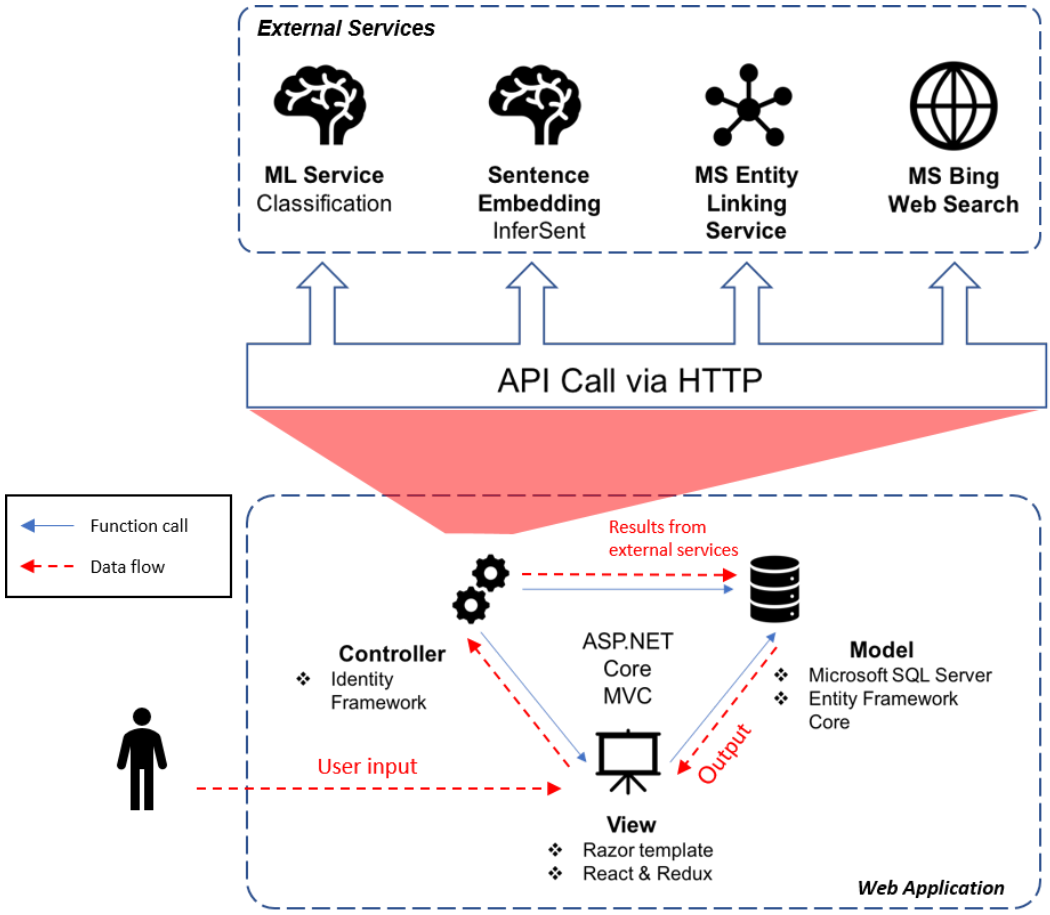

Figure 10. Final application architecture.

4.2.2 Database Design

We created a relational database to efficiently store information for our system such as user data or text entries submission.. The database schema is illustrated in Figure 10. There are four entities in the database schema: User, TextEntry, and Sentence. Each user in the database corresponds to a user account. Users are differentiated by unique ids. Additionally, each user can submit multiple text entries. In this case, text entries are defined as the entire piece of text a user submits. Text entries are also differentiated by unique ids. On top of this, they have other attributes: Content, CreatedAt, and UserId (the ID of the user who submitted the text entry). Moreover, each text entry can contain multiple sentences, which also have unique ids as the

primary key. Other attributes include VoteFalse (number of downvotes), VoteTrue (number of upvotes), Confidence (how certain the sentence classification is), Type (sentence category: objective/subjective/other), TextEntryId (the ID of the original text entry which the sentence is part of), Content, Position (the order of the sentence in the text entry), and InferSentVectorsString (the sentence embedding vector computed by InferSent). For the Vote table, there are two primary keys: SentenceId, and UserId. These are also foreign keys. SentenceId links the Vote table to the Sentence table. UserId links the Vote table to the User table. The Vote table also has timestamp, which records the time at which the vote is made.

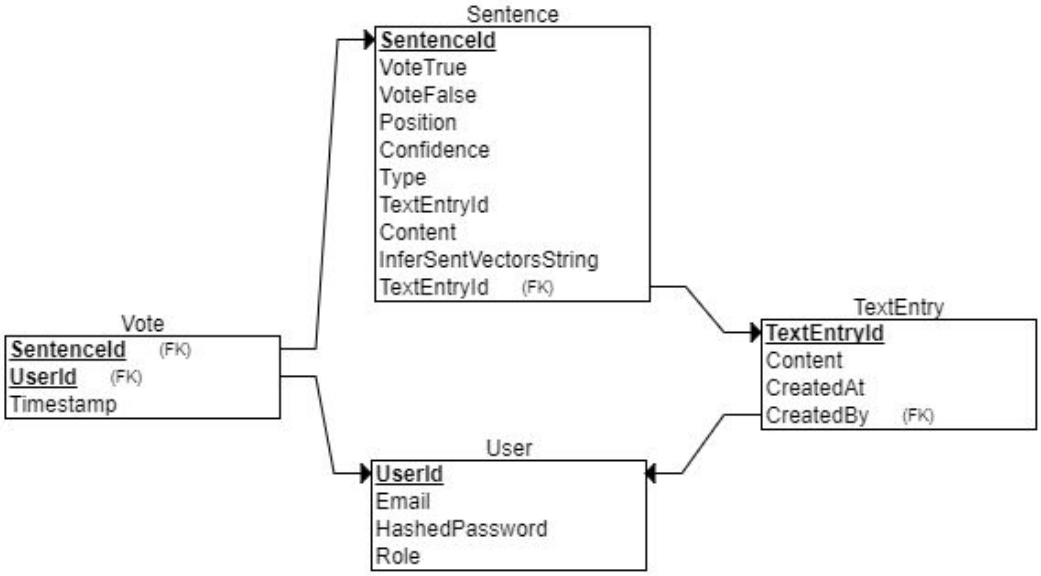

Figure 11. Relational database diagram

4.2.3 Web Controllers

ASP.NET Core enabled us to create controllers effortlessly with an MVC Controller template. These controllers manage the data that come from user interactions on the web interface and interact with databases to update the information.

- **Account**: The Account controller manages user registration and login,
- **TextEntry**: The TextEntry controller receives user input and calls the parser to tokenize the entry into individual sentences.
- **Sentence**: The Sentence controller has three endpoints: Feed, Details, and Related. The Feed endpoint handles fetching latest sentences submitted in reverse chronological order. The Details endpoint calls the Bing Entity Linking service and returns entities extracted from a particular sentence and related links from Bing Search Engine. The Related endpoint calls our machine learning service that computes sentence similarity and returns at most five sentences most related to a selected sentence.

- **Vote**: The Vote controller manages the number of upvotes and downvotes for each sentence in the feed.

4.2.4 Machine Learning Services

We used Azure Machine Learning to deploy the machine learning service ("ML Service" in Figure 9) and the sentence embedding service ("Sentence Embedding" in Figure 9). The Azure Machine Learning is an integrated, end-to-end data cloud framework that enables data scientists to prepare data, develop experiments, and deploy models.

The specific libraries we used to build the services and models detailed above in Chapter 3 are as follows. We use NLTK (Natural Language Toolkit) for parsing a piece of text into sentences. NLTK is the leading platform for building programs to work with human language in Python. Although NLTK provides numerous capabilities such as tagging text and extracting entities, we only use it to tokenize text into sentences. For the similar sentences features, we used InferSent to embed the sentences. We used the Python library scikit-learn to train the shallow machine learning classifiers such as Naive Bayes and Support Vector Machines. We used CNTK to train the deep learning models.

4.3 Front-end

In order to provide the desired features to our users, we designed and programmed a web interface. In this section, we refer specifically to the design of the dynamic front-end web interface.

The structure and data flow of our interface were built complying with the principles imposed by the React and Redux libraries. These principles allow for composable and flexible components. Furthermore, when using React with a structured flow of data, in our case facilitated using Redux, it is very straightforward to reason about, implement, and reuse. We outline the structure of the application in Figure 11. In short, each React component has to be designed as an immutable tree of HTML elements or other components. All components can be traced back to a root component; in our diagram, this is referred to as 'React component tree'. On the other hand, the Redux principle ensures the flow of data to be one-directional. When an event is triggered on the React component tree, an action object is dispatched. After that, the reducers in charge of the action are run to alter the state of the interface. This state is the only source of data throughout the whole interface. Finally, the changes in the state trigger new rendering of the React component tree. Data extracted from the state will be trickled down from the root to all nodes in this tree.

The primary purpose of the Fact Fiction web interface is to showcase the capabilities of the sentence classification technology in a clear and complete manner. It has a single view; a form

that allows a user to paste textual content. Then user receives contextual information that expands upon user interaction and interest. The visual results of our design are presented in Section 4.4.

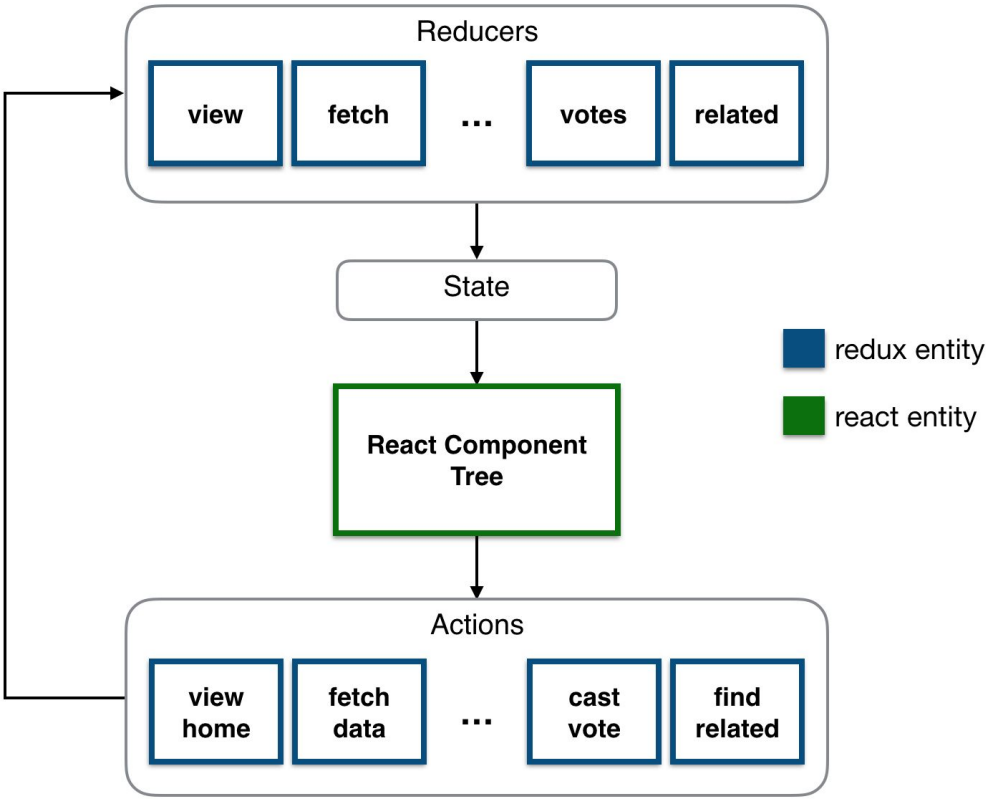

Figure 12. Fact or Fiction Application front-end structure

4.4 Fact or Fiction Application User Interface

The team built a scalable website hosted on Azure Web services with a clean and friendly interface using React, Redux, and Office Fabric UI. The interface includes four different views:

- Welcome screen (Figure 13)
- Login screen (Figure 14)
- Text input and feed view (Figure 15)
- Result view (Figure 16)

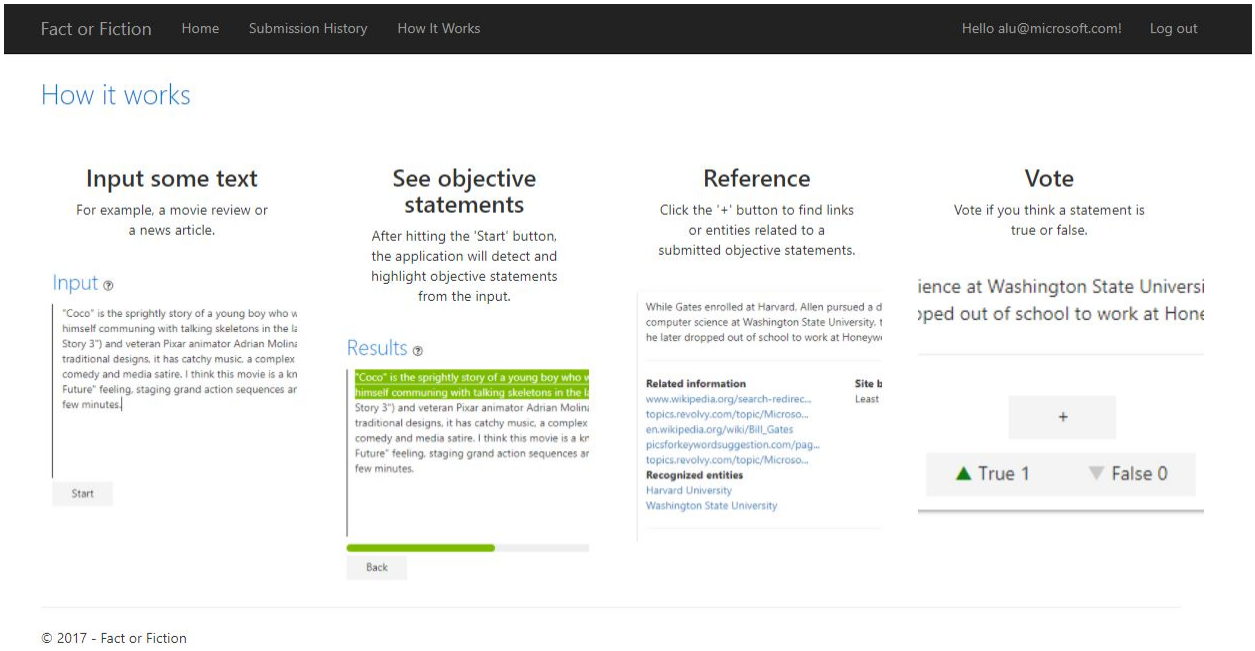

Figure 13. Fact or Fiction Application Welcome screen.

When a user is not logged in, a welcome page shows brief introduction about the features on Fact or Fiction, including text input, objective sentences highlight, reference, and vote. This screen aims to help new users understand what Fact or Fiction is and how to use it.

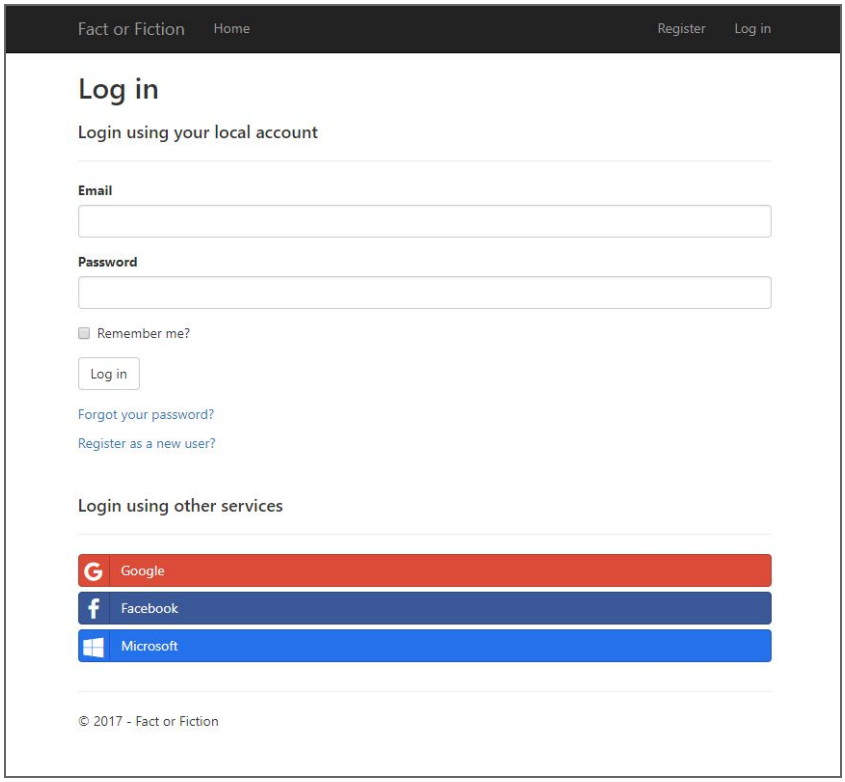

Figure 14. Login screen.

The user would be directed to this Login screen once they click on "Login" on the upper right corner of the navigation bar. Our website supports signing up by using their email address that they previously had as well as by using other external services such as Google, Facebook, and Microsoft accounts. We had two consideration in mind by providing both external services and email registration:

1. The external services linking reduces the number of steps required for account registration and makes it easier for users to manage their account.
2. The in-house email registration gives users more flexibility in choosing which email account to use and makes it available to users who do not have any of the listed services.

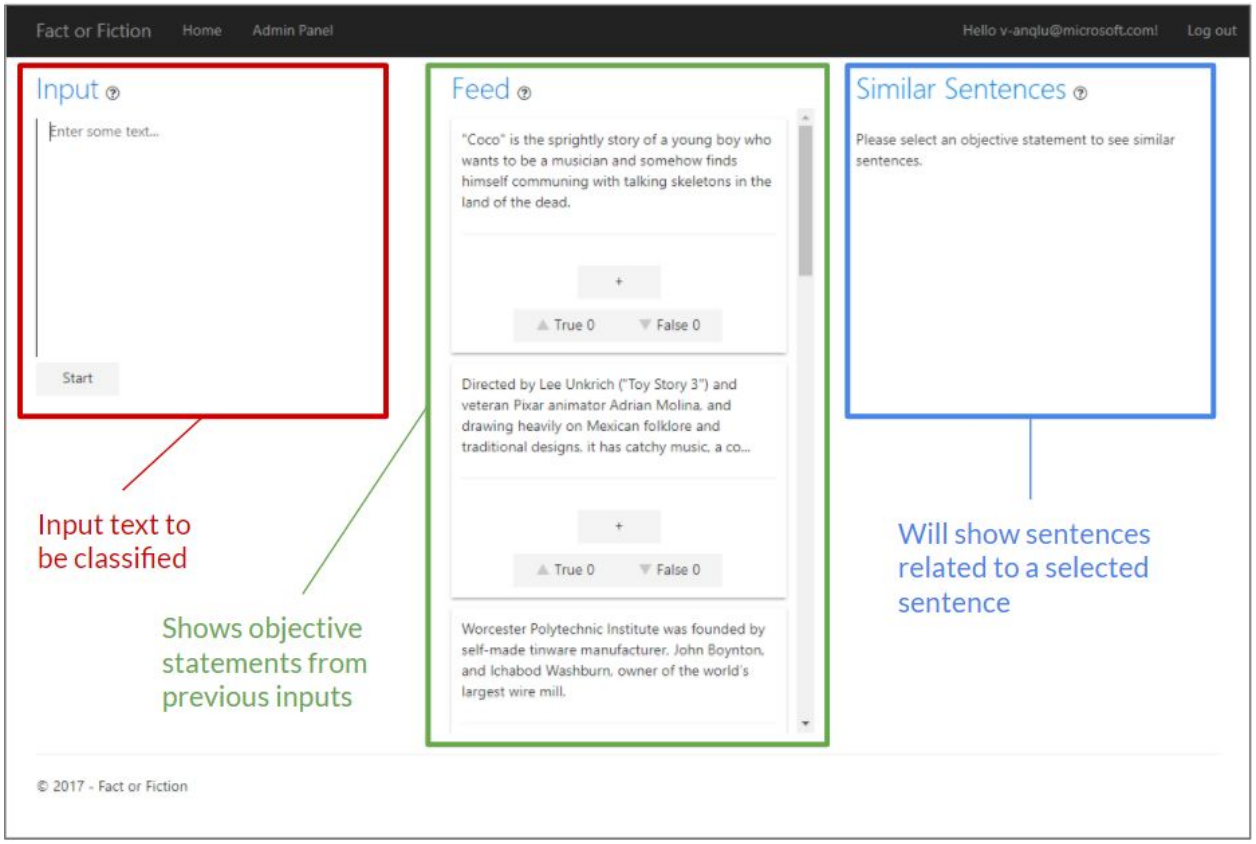

Figure 15. Text input and feed view.

Once logged in, a user sees home view shown in Figure 15, which has three columns:

- **Input**: where users copy and paste or input a block of text they would like to analyze in the input box.
- **Feed**: objective sentences extracted from other users input, sorted in descending order by the time of input. Once a user scrolls to the bottom, the feed will load more objective sentences. Each objective sentence is presented in a card that shows votes others have cast and can show more details by clicking on the "+" button, illustrated in Figure 16.
- **Similar Sentences**: Once a user selects an objective sentence card, the similar sentences column will display a maximum number of five other objective sentence cards that are the most related from the previous submissions of all users. This functionality is illustrated in detail in Figure 17.

The question mark icon next to the column titles are tooltips that give a brief introduction about each column. The explanation shows once a user hovers the mouse over the icon. When the user finishes inputting text, clicking the "Start" button will bring the user to the result view shown in Figure 16.

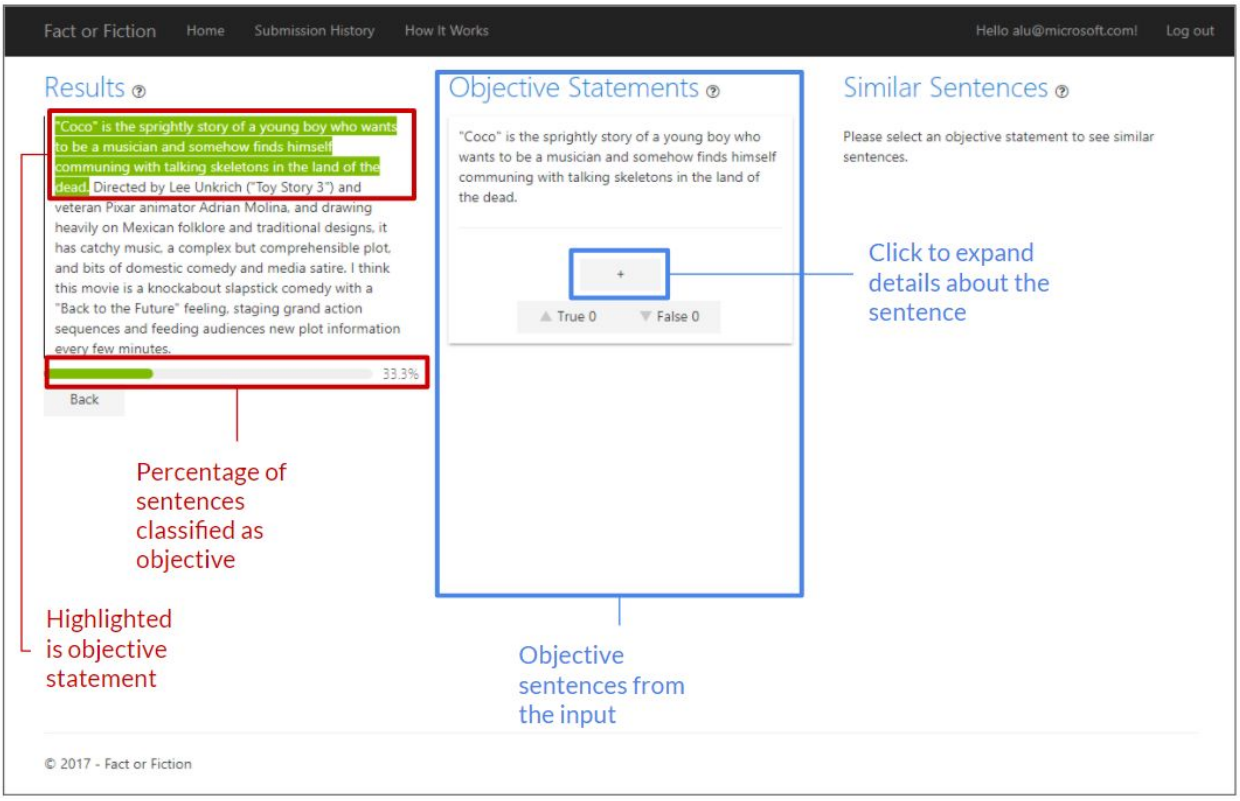

Figure 16. The Result view.

The Result view (shown in Figure 16) also contains three columns:

- **Results**: This column shows the classification results by highlighting objective sentences in green. A user can hover and select a sentence to see the corresponding objective sentence card and the card automatically scrolls into the view. The bar below the text shows the percentage of objective sentences within the text input.
- **Objective Statements**: The cards in this column (left side of Figure 17) corresponds to each objective sentence highlighted in the Results column.
- **Similar Sentences**: These are sentences that are similar to any selected objective sentence (right side of Figure 17).

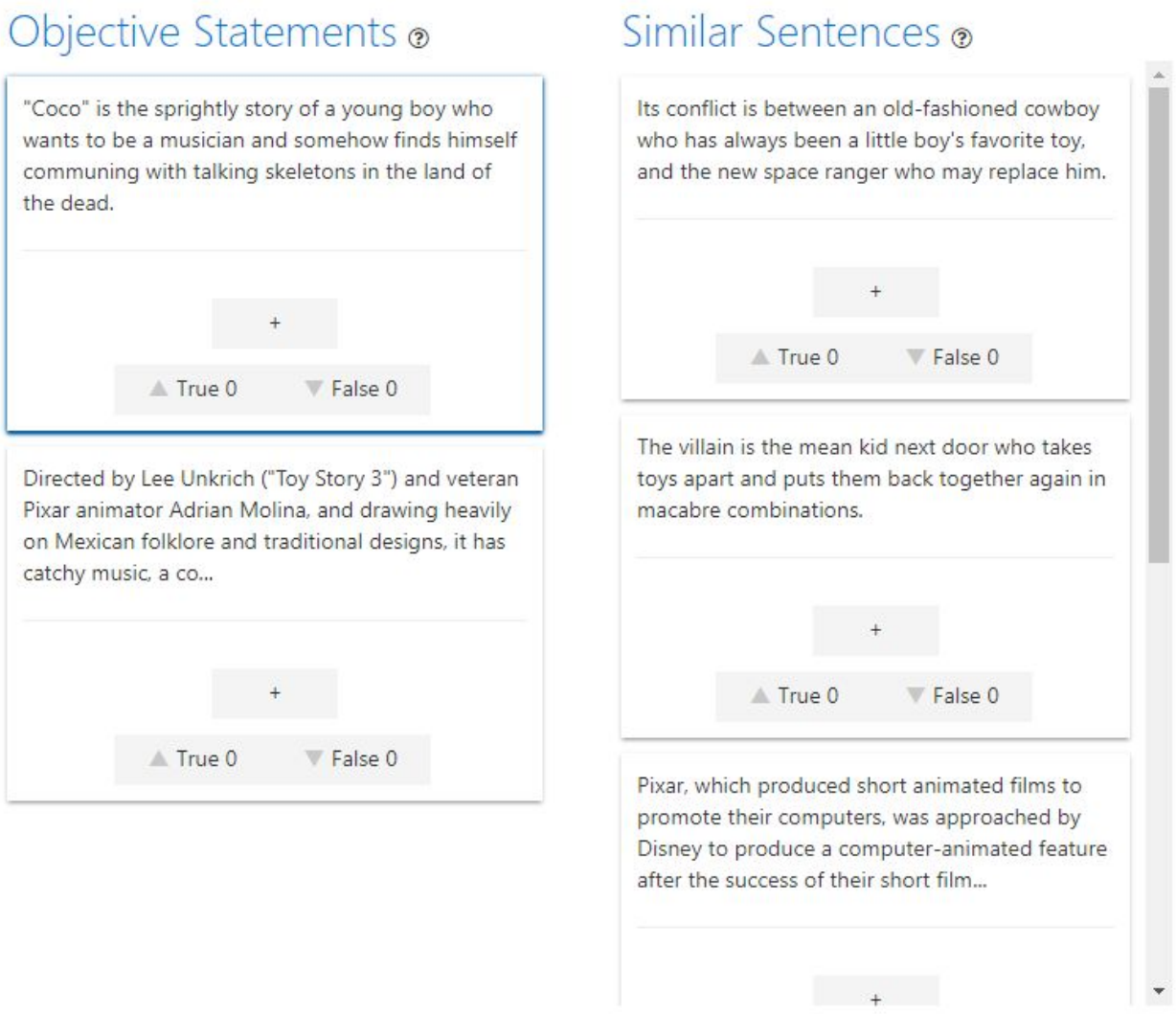

Figure 17. The Similar Sentences Feature.

When a user clicks on a card, the card is selected and is highlighted with blue borders. Our server then fetches at most five other objective sentences from our database, which are most related to the selected one. As shown in Figure 17, the similar sentences do not necessarily share the similar words syntactically; instead, it analyzes the intent of the sentence and returns the sentences with similar intentions. In Figure 16, the selected sentence is about the Pixar animated film, *Coco*, and the similar sentences shown are about the plot of *Toy Story* and about Pixar.

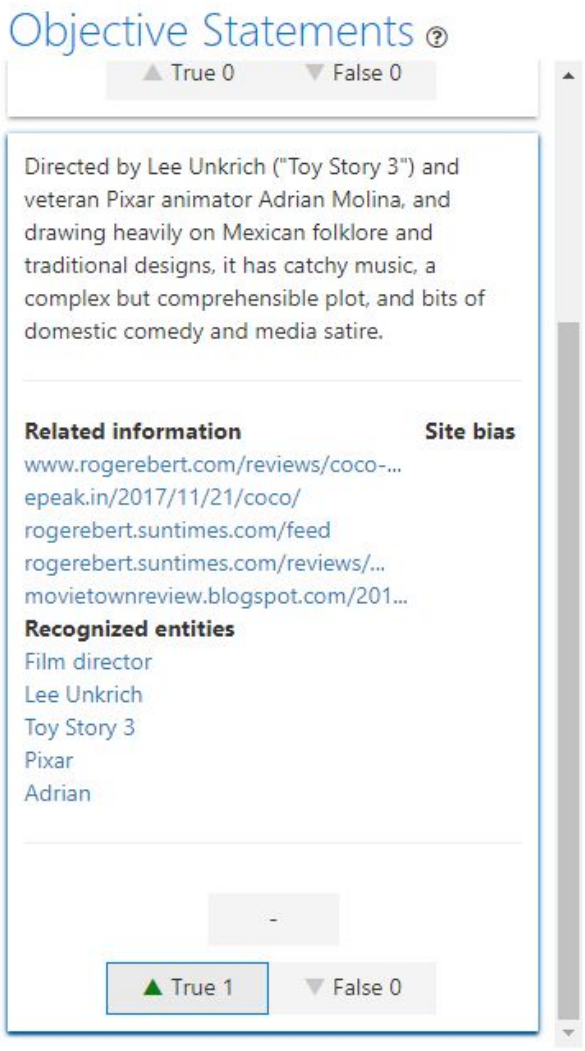

Figure 18. The Details, Entity Extraction, and Vote Features.

Within each objective sentence card, a user can get more information about the sentence by clicking on the "+" button directly below. Once clicked, the card expands and shows a table as shown in Figure 18. The "Related information" section provides links that are related to the sentence. The "Recognized entities" section shows entities that are extracted from the sentence. If available, "Site bias" would show potential bias from links provided. The user can also click on the "-' button to hide this information.

Since our model only classifies whether a sentence is objective or not, it does not tell if a sentence is actually true or false. It is up to users as a community to fact-check the sentences. By using their prior knowledge and the references link provided by Fact or Fiction, users can determine the veracity of sentences. After that, they can also vote the sentences to be true or false

through the buttons. One user can only vote once on the same sentence. Cancelling and changing vote are also allowed.

We are satisfied with the technologies we chose in developing the web interface. Office Fabric UI enabled us to create a clean user interface with a style that is consistent with Microsoft's products. React helped us manage different components of the websites that are flexible and extensible. Redux made it simple and clear to manage the changing states when actions are triggered by users.

Chapter 5. Results

In this chapter, we present the results of our implementation and experimentation that follows the methodologies in chapter 3. In section 5.1, we show the results of our experiment on the machine learning classification models with plausible explanations. In section 5.2, we demonstrate the final application and the technologies involved in the implementation. In section 5.3, we report the results of the user experience surveys and interviews.

5.1 Machine Learning Classification Results

Table 4 shows the results of the two machine learning (Naïve Bayes and SVM) and two deep learning (LSTM and CNN) models, which were evaluated using 3 different metrics: AUC, accuracy, and F1-score. These metrics were previously explained in detail in section 3.1.4.

Table 4. Classification performance results

Models	AUC	Accuracy	F1-score
Long Short Term Memory (LSTM)	0.9023	82.89%	0.8162
Convolutional Neural Network (CNN)	**0.9332**	**85.69%**	**0.8538**
Naive Bayes	0.8409	84.18%	0.8328
Support Vector Machine	0.8468	84.66%	0.8434

One observation here is that both shallow models (SVM and Naïve Bayes) and the convolutional neural network both outperform the Long Short Term Memory (LSTM) model. This might be explained by the fact that when humans determine whether a sentence is subjective or objective, they usually look at some signal words or phrases scattered throughout a sentence rather than examining the whole sequence as a whole.

The LSTM model performs classification by learning the long-term dependency of the whole sequence. For example, when the LSTM model examines the sentence “Book me a flight to New York.”, it recognizes the entity “New York” at the end of the sentence and remembers the word “Book” at the beginning of the sentence, which indicates an interest in scheduling a flight.

However, CNN looks at different segments of a sentence and Naive Bayes and Support Vector Machine (SVM) looks at discrete feature values of signals such as the phrase “I think” or “In my opinion” to classify the sentences, which is closer to what we humans do. As a result, we

deployed the CNN model to our service because it scored the highest in both accuracy and F1-score.

When tuning hyperparameters for Naive Bayes and SVM models, we used grid search to look for the best parameters for each model. For both models, we tried three sets of Part of Speech tagging (POS):

- All types of adjectives, all types of verb, all types of adverb, all types of nouns
- Just all types of adjective
- None

The results show that all three sets of POS tagging have the same mean test scores of 0.8378. Therefore, the POS tagging feature did not prove to be a predictive feature in the task of classifying objective and other sentences.

For the SVM model, we tried both linear and RBF kernels. For RBF kernels, we further tried gamma values of 0.001 and 0.0001. The linear kernel had a mean test F1-score of 0.8375 whereas RBF kernel for both gamma values only had a mean F1-score of 0.5157 for the test set. It is unclear why the RBF kernel had such bad performance. We speculate that it has to do with the relatively large number of features, compared with the number of training examples. The number of features used in for both the linear and RBF kernels was directly proportional to the size of our vocabulary, ~100,000, and was applied to each td-idf n-gram and objectivity score, yielding a total number of features which was greater than our ~12,000 training examples. Unlike the deep models that embed inputs into a dense vector space, shallow models work directly in this high-dimensional feature space. A simpler model, the linear kernel, trained faster than the more complex RBF kernel and was easier to tune.

5.2 Performance of Back-end Services

Since our system was deployed as a working system, it was important to evaluate real-time performance numbers such as end-to-end response time. A system that is too slow would frustrate users, who likely would abandon it. Microsoft Visual Studio Team Services provides various types of testing, including load testing measurements. The load tests feature uses Visual Studio Enterprise to create a large volume of synthetic HTTP requests to send to our server, to see how well our server respond to these requests. In our load testing, we tested five endpoints, corresponding to five major features of the Fact or Fiction application:

- Vote: users cast a true or false vote on a particular sentence
- Text Entry: users post text for its sentences to be classified as objective or subjective
- Related: Retrieve similar sentences for a particular sentence
- Feed: Fetching objective sentences from the database

- Reference: Retrieving related links for a particular sentence

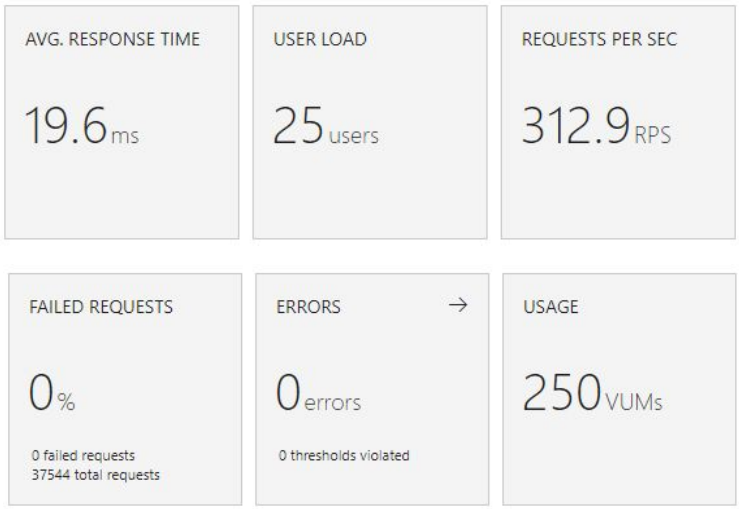

Figure 19. Load test metrics.

The load tests had six metrics.

- *Average response time:* the average time it took for our app to respond to requests.
- *User load:* The peak user load encountered during the load test run.
- *Requests per second:* number of requests sent to our app per second.
- *Failed requests:* percentage of requests failed.
- *Errors:* the number of failed requests from the total sent requests.
- *Usage:* an estimate of how many minutes of user load spent on service.

As shown in Figure 19, the load test sent 312.9 requests per second to our service. Our server handled all 37,544 requests with no error and the average response time for each request was 19.6 milliseconds.

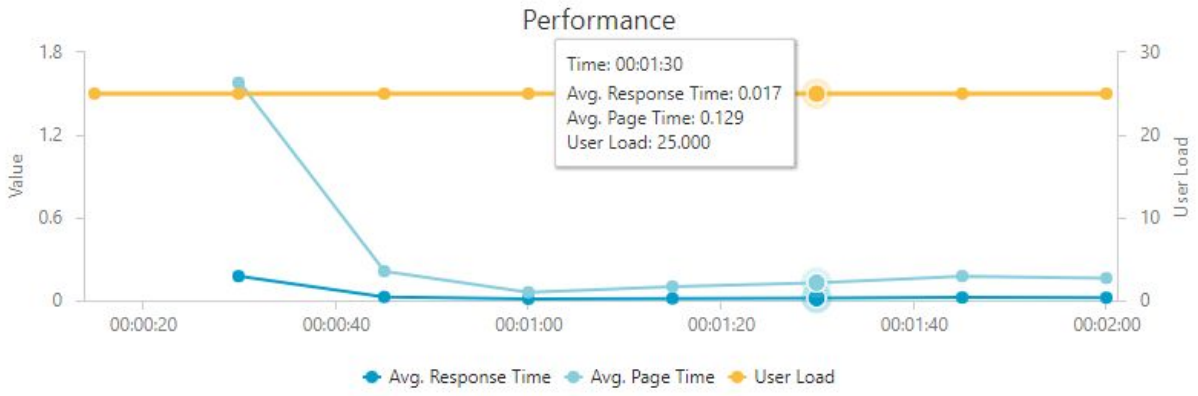

Figure 20. Line chart of Performance of Fact or Fiction Web Endpoints.

The testing framework in Visual Studio Team Services also provides a visualization of the performance over time in line charts. Figure 20 illustrates the system performance for a span of two minutes. The darker blue line represents average response time; light blue line represents average page loading time; and the yellow line represents user load, number of users this load testing mimics to send requests to our website. All three lines share the x-axis, whereas the two blue lines correspond to the y-axis on the left (Value) and yellow line corresponds to y-axis on the right (User Load). The load test synthetically mimics 25 users sending requests to our server, and so the yellow line remains 25. Our response time is 0.017 seconds at time 01:30 as shown in the pop-up tooltip displayed. The trend of the line graph shows that our server took longer initially and then its response time decreased drastically and subsequently remained fairly stable.

5.3 User Experience Results

Our final evaluations pertained to benchmarking the user's experience while using our Fact or Fiction website. We first compute an average score for each of the eight questions which aim to measure the user experience (see Section 3.3.1). Then we summed average scores for the two questions from each of the four categories to get four average scores for the corresponding factors. These factors included *usability*, *credibility*, *loyalty*, and *appearance*. For each factor, the minimum score is 2 and maximum is 10. We determined each factor to have been met if it receives a score of 6 or higher as the average user would find that factor satisfactory. We considered our website excellent for a given factor if it scores 8 or higher. We also summed all the average scores up to get a total score. The minimum score possible was 8 and the maximum was 40. Overall, we considered that our website usable if it scores a total score of 24 or higher. We considered the whole website satisfactory if it scores 32 or higher.

5.3.1 Survey Results

We conducted an anonymous survey to evaluate the user experience of our platform. We posted the survey link on WPI student Facebook group as well as sending it to acquaintances of the team members. On the results were positive, with the usability and the appearance scoring highly (see Table 5 below). However, we had mixed results when prompting people whether or not they were likely to recommend the website or return to the site themselves. This contrasts with the usability and credibility feedback of the site, and in our analysis increase the likelihood that this website is an effective tool for showcasing the machine learning models we built, but not the perfect medium for using them. As suggested in our open-ended interviews (see Section 5.3.2 below), two possible solutions to this are to expose our services as an API and to create a web plugin.

The average scores for each question are recorded in the table below (Table 5. User Experience Survey Results). Three of the user experience factors, usability, credibility, and appearance, were satisfactory according to our guidelines as their responses all averages to above 6 at 7.79, 7.53, and 7.06. None of these were above 8, and thus none were excellent according to our guidelines, as outlined in the beginning of this section. Lastly, loyalty was not satisfactory because it was less than 6 at 5.53. The relative low loyalty score indicates that the users did not see the need for themselves and others to use our website.

Table 5. User Experience Survey Results

Factors	Questions	Average Score of 20 Responses (1-5) 1 = Strongly disagree 5 = Strongly agree	Average Score for each factor
Usability	The website is easy to use.	3.84	7.79
	It is easy to navigate within the website.	3.95	
Credibility	The information on the website is credible	3.79	7.53
	The information on the website is trustworthy.	3.74	
Loyalty	How likely are you to recommend this website to a friend or colleague?	2.95	5.53
	I will likely return to the website in the future.	2.58	
Appearance	I find the website to be attractive.	3.11	7.06
	The website has a clean and simple presentation.	3.95	

5.3.2 Qualitative Interview Results

We interviewed 5 Microsoft employees in order to receive more qualitative feedback regarding the user experience beyond the quantitative data from the survey. These interviews were held in-person. The interviews were successful, providing us with both low-level fixes and high-level suggestions. Feedback during the surveys is listed and organized in Tables 6-10 below.

In general, the usability feedback was useful, as it clearly indicated some strong points of our work, and some things to improve upon. These comments, noted in Table 6, are that the website itself was easy to use with a clean presentation, but the user flow could be improved upon. Minor additions like adding a link to the help-page, increasing the prominence of the input section, and better indicating the order of actions would help users better understand what actions to take first.

The feedback for credibility, trustworthiness, and loyalty was direct (Tables 7-9). The only constructive criticism here was that the subject wanted to see more features and have our project continue.

Finally, we found the general comments (Table 10), extremely useful in identifying future directions for our work. Namely, extending the underlying services we created, to allow other developers and sites to use them directly. This was in line with our internal evaluation of the project. We had considered the browser plugin and exposing the API, but chose to consider it as future work due to time constraints. The users were in general highly interested in the site and would come back to try it out again in their free time. They did note that keeping the target audience in mind would help the development process of the UI so that it would be more intuitive.

We, as observers of the interviews, also note that the users were quick to ignore details from the help page and usually did not use the tooltips to learn more. This makes it even more important for us to ensure the website is self-explanatory, without relying heavily on a help page that most users would not ultimately reference.

Table 6: Usability Feedback

Usability	
Positive	Constructive
It is easy to navigate within the website.	It is not obvious how to go to the homepage from the help page.
The website is easy to use.	Increase the prominence of the input section.
Laid out well - left to right - guide people through	Change the layout of the help screen, to better indicate the flow of user actions.
The website has a clean and simple presentation.	The expand and detract buttons aren't clear.
The resizing for different screen size layouts was intuitive.	Highlighting of objective sentences is good, but a legend would help.

Table 7: Credibility Feedback

Credibility	
Positive	Constructive
The information on the website seems credible, the classification works, and the references all make sense.	-

Table 8: Trustworthiness Feedback

Trustworthiness	
Positive	Constructive
The information on the website is trustworthy.	-

Table 9: Loyalty Feedback

Loyalty	
Positive	Constructive
I will likely return to the website in the future.	I'd be very interested to see new features!

Table 10: Constructive Feedback

Further Constructive Comments
Adding a comment feature to the statements would be awesome!
Expose your service as an API.
Web browser plugin to highlight on the news articles as one reads.

Chapter 6. Conclusion and Future Work

6.1 Conclusion

Polarization and fake news are increasingly pervasive online. Several studies have found the effects of these phenomena detrimental to the public good (Allcott & Gentzkow, 2017; Barthel et al., 2016). A purely technological solution to these problems is intractable, but creating platforms and support tools to better guide readers' judgment and provide rich resources will have a positive impact. For this project, we built a web platform to foster constructive, objective discussion based on data. To do this, we built a performant model which extracts objective sentences from a document. When a user inputs text, the platform extracts and highlights the objective sentences and presents them directly to the user. This allows users to fact-check statements with external resources. We further supplement this feature by providing links to reliable references and sources regarding those statements. We deployed this application using ASP.Net Core and Azure Cloud services. Our machine learning models were deployed using Azure Machine Learning Workbench. This makes our application scalable and modular. The performance of our machine learning models is ~85% for a convolutional neural network (CNN). We found that the shallow models, Support Vector Machine (SVM) and Naïve Bayes, had similar performance, whereas, the recurrent neural network (RNN) did not perform as well (see Section 5.4.1). We postulate that more data is necessary to fully utilize this more complex model. The qualitative results from both the survey and our open-ended interviews indicated that the website was a clean and useful application, and importantly effectively highlighted a clear path for future work to extend the underlying methods we developed to external applications.

6.2 Future Work

Expose Our Service as an API - It would be beneficial to expose our service as an API that takes in a chunk of text and returns structured data that contains the classification of the sentences and other detailed information. This would enable other developers to use our service to create other applications and potentially increase the influence of this project.

Web Browser Plugin - Ultimately, integrating this pipeline into the media habits of readers will allow them to read in a more informed manner. For example, equipping readers with the knowledge of what is factual, what is conjecture, what is intentionally inflammatory and linking known entities to trusted sources of information would help readers to quickly parse through information. This would not only negate fake news, it would lead to more informed readers.

Comment Feature on the Website - As David Hurley imagined when he started this project, he would like Fact or Fiction to be a platform where people can have sensible discussions. A

comment section underneath each objective sentence in the current design would enable users to discuss current events based on just facts.

Deploy in More Languages - We envision a system that would work seamlessly across languages and localities, capable of globally helping readers to discern for themselves a more reasoned basis of fact and fiction. This is difficult as our machine learning model requires training data to work in any given language, but we postulate that using machine translation could help solve this problem.

Improve the Machine Learning Model - In order to improve performance and effectively use deep learning without overfitting the network, we see the need to expand on the dataset we have to cover more diverse grammar, syntaxes, and styles of writing. We can also try ensemble learning (e.g. bagging or boosting), which is a method that combines several weak learners to make more accurate classification decisions.

References

Abramowitz, A. I., & Saunders, K. L. (1998). *Ideological realignment in the US electorate*. The Journal of Politics, 60(3), 634-652.

Allcott, H., & Gentzkow, M. (2017). *Social media and fake news in the 2016 election* (No. w23089). National Bureau of Economic Research.

Aggarwal, C. C., & Zhai, C. (2012). A survey of text classification algorithms. Mining text data, 163-222.

Armstrong, Martin. (2016, November 17). *Fake News Is A Real Problem*. Retrieved December 12, 2017 from https://www.statista.com/chart/6795/fake-news-is-a-real-problem/.

Artificial Neural Network (ANN). (n.d.). Retrieved October 21, 2017, from https://www.techopedia.com/definition/5967/artificial-neural-network-ann

Bartels, L. M. (2000). *Partisanship and voting behavior, 1952-1996*. American Journal of Political Science, 35-50.

Barthel, M., Mitchell, A., & Holcomb, J. (2016) *Many Americans Believe Fake News Is Sowing Confusion.* Pew Research Center's Journalism Project. Retrieved December 8, 2017 from http://www.journalism.org/2016/12/15/many-americans-believe-fake-news-is-sowing-confusion

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005, August). Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning* (pp. 89-96). ACM.

BuzzFeedNews Github. (n.d.). Retrieved from https://github.com/BuzzFeedNews/everything#data-and-analyses

Conger, Kate. (2017, February 8). *John Podesta talks email hack, fake news and Russia*. Retrieved December 12, 2017 from https://techcrunch.com/2017/02/08/john-podesta-talks-email-hack-fake-news-and-russia/.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., & Bordes, A. (2017). Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. *arXiv preprint arXiv:1705.02364*.

Fernández, S., Graves, A., & Schmidhuber, J. (2007). An application of recurrent neural

networks to discriminative keyword spotting. Artificial Neural Networks–ICANN 2007, 220-229.

Frakes, W. B., & Baeza-Yates, R. (1992). *Information retrieval: data structures and algorithms*.

Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning* (Vol. 1, pp. 241-249). New York: Springer series in statistics.

Gers, F. A., Schmidhuber, J., & Cummins, F. (1999). *Learning to forget: Continual prediction with LSTM*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Google (2013, July 29). *word2vec*. Retrieved December 12, 2017 from https://code.google.com/archive/p/word2vec/

Gray, Richard. (2017, March 1). *Lies, propaganda and fake news: A challenge for our age*. Retrieved October 16, 2017 from http://www.bbc.com/future/story/20170301-lies-propaganda-and-fake-news-a-grand-challenge-of-our-age.

Greenbaum, S., Quirk, R., Leech, G., & Svartvik, J. (1985). *A comprehensive grammar of the English language*. USA: Longman. Retrieved November, 4, 2012.

Greene, D., & Cunningham, P. (2006, June). *Practical solutions to the problem of diagonal dominance in kernel document clustering*. In Proceedings of the 23rd international conference on Machine learning (pp. 377-384). ACM.

Guynn, J. (2017, March 07). Facebook begins flagging 'disputed' (fake) news. Retrieved October 20, 2017, from https://www.usatoday.com/story/tech/news/2017/03/06/facebook-begins-flagging-disputed-fake-news/98804948/

Hatzivassiloglou, V., & Wiebe, J. M. (2000, July). *Effects of adjective orientation and gradability on sentence subjectivity.* In Proceedings of the 18th conference on Computational linguistics-Volume 1 (pp. 299-305). Association for Computational Linguistics.

Hetherington, M. J., & Rudolph, T. J. (2015). *Why Washington won't work: Polarization, political trust, and the governing crisis*. University of Chicago Press.

Hochreiter, S., & Schmidhuber, J. (1997). *Long short-term memory*. Neural computation, 9(8), 1735-1780.

Infersent Github. (n.d.). Retrieved December 6, 2017 from https://github.com/facebookresearch/InferSent.

K., I., & K., S. (2017, January 19). *A flood of false headlines probably did not swing America's election*. Retrieved October 7, 2017 from https://www.economist.com/blogs/freeexchange/2017/01/fake-news.

Kim, Y. (2014). *Convolutional neural networks for sentence classification*. arXiv preprint arXiv:1408.5882.

Kolodny, L. (2016, November 17). Full Fact aims to end fake news with automated fact checking tools. Retrieved October 18, 2017, from https://techcrunch.com/2016/11/17/full-fact-scores-google-grant-to-put-an-end-to-fake-news/

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *Imagenet classification with deep convolutional neural networks.* In Advances in neural information processing systems (pp. 1097-1105).

Lardieri, Alexa. (2017, September 20). *Fake News, Real Impact*. Retrieved October 8, 2017 from https://www.usnews.com/news/national-news/articles/2017-09-20/team-lewis-study-examines-how-fake-news-impacts-public.

Lelkes, Y. (2016). *Mass polarization: Manifestations and measurements*. Public Opinion Quarterly, 80(S1), 392-410.

Li, David. (2016, March 28). *Entity Linking Intelligence Service API*. Retrieved December 8, 2017 from https://docs.microsoft.com/en-us/azure/cognitive-services/entitylinking/home.

LUIS Homepage. (n.d.). Retrieved December 1, 2017 from https://www.luis.ai.

Maaten, L. V. D., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*(Nov), 2579-2605.

Microsoft Azure. (n.d.). Retrieved December 1, 2017 from https://azure.microsoft.com/en-us/.

Microsoft Azure Region Map. (n.d.). Retrieved December 13, 2017 from

http://map.buildazure.com/

Microsoft Cognitive Toolkit Features. (n.d.). Retrived December 12, 2017 from https://www.microsoft.com/en-us/cognitive-toolkit/features/

Microsoft Cognitive Toolkit Github. (n.d.). Retrieved December 11, 2017 from https://github.com/Microsoft/CNTK/.

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010, September). *Recurrent neural network based language model*. In Interspeech (Vol. 2, p. 3).

Moon, M. (2017, October 24). Facebook's new media guidelines are focused on stopping fake news. Retrieved October 30, 2017, from https://www.engadget.com/2017/10/24/facebooks-new-media-guidelines-stopping-fake-new/

Mou, L., Li, G., Zhang, L., Wang, T., & Jin, Z. (2016, February). Convolutional Neural Networks over Tree Structures for Programming Language Processing. In AAAI (pp. 1287-1293).

MPQA Corpus Release Page. (n.d.). Retrieved October 12, 2017 from http://mpqa.cs.pitt.edu/corpora/mpqa_corpus/.

Office of the Director of National Intelligence. (2017, January 6). *Background to "Assessing Russian Activities and Intentions in Recent US Elections": The Analytic Process and Cyber Incident Attribution*. Retrieved October 8, 2017 from https://www.dni.gov/files/documents/ICA_2017_01.pdf.

Olah, C., Mordvintsev, A., & Schubert, L. (2017). *Feature Visualization*. Distill, 2(11), e7.

Pang, B., & Lee, L. (2004, July). *A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts*. In Proceedings of the 42nd annual meeting on Association for Computational Linguistics (p. 271). Association for Computational Linguistics.

Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, *2*(1–2), 1-135.

Qian, M., & Zhai, C. (2014, November). *Unsupervised feature selection for multi-view clustering on text-image web news data*. In Proceedings of the 23rd ACM international conference on conference on information and knowledge management(pp. 1963-1966). ACM.

Rapoza, Kenneth. (2016, December 30). *Wrong Again: Russia's Anglo-American School Not Closing To Spite Obama*. Retrieved October 8, 2017 from

https://www.forbes.com/sites/kenrapoza/2016/12/30/wrong-again-russias-anglo-american-school-not-closing-to-spite-obama/#2321e01f50d8.

React. (n.d.). Retrieved December 12, 2017 from https://reactjs.org.

Reese, S., Boleda, G., Cuadros, M., & Rigau, G. (2010). *Wikicorpus: A word-sense disambiguated multilingual wikipedia corpus*.

Redux. (n.d.). Retrieved December 12, 2017 from https://redux.js.org/.

Sauro, J. (2015). SUPR-Q: *A comprehensive measure of the quality of the website user experience*. Journal of usability studies, 10(2), 68-86.

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

Stepinski, A., & Mittal, V. (2007, July). *A fact/opinion classifier for news articles*. In Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval (pp. 807-808). ACM.

Techopedia. (n.d.). Retrieved December 12, 2017 from https://www.techopedia.com.

The Economist, *Ukraine's reform activists are under attack*. (2017, August 24). Retrieved October 7, 2017 from https://www.economist.com/news/europe/21727110-fake-news-threats-and-arrests-corrupt-system-fighting-back-ukraines-reform-activists.

Wang, T., Wu, D. J., Coates, A., & Ng, A. Y. (2012, November). *End-to-end text recognition with convolutional neural networks*. In Pattern Recognition (ICPR), 2012 21st International Conference on (pp. 3304-3308). IEEE.

Webb, Alex. (2017, October 2). *Facebook Says 10 Million People Saw Russia-Linked Advertisements*. Retrieved October 7, 2017 from https://www.bloomberg.com/news/articles/2017-10-02/facebook-says-10-million-people-saw-russia-linked-advertisements.

Wiebe, J. M., Bruce, R. F., & O'Hara, T. P. (1999, June). Development and use of a gold-standard data set for subjectivity classifications. In Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics (pp. 246-253). Association for Computational Linguistics.

Wilson, T., Hoffmann, P., Somasundaran, S., Kessler, J., Wiebe, J., Choi, Y., ... & Patwardhan, S.

(2005, October). *OpinionFinder: A system for subjectivity analysis*. In Proceedings of hlt/emnlp on interactive demonstrations (pp. 34-35). Association for Computational Linguistics.

Wilson, T., & Wiebe, J. (2003). *Annotating Opinions in the World Press*. In SIGDIAL Workshop (pp. 13-22).

Yu, H., & Hatzivassiloglou, V. (2003, July). *Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences*. In Proceedings of the 2003 conference on Empirical methods in natural language processing (pp. 129-136). Association for Computational Linguistics.

Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. arXiv preprint arXiv:1212.5701.