Bilkent University

Department of Computer Engineering

CS 461 - Artificial Intelligence

Term Project Report

FIRING JOEL FAGLIANO

Project Group Members:

Barış Furkan Ünsal	21502062
Çağatay Küpeli	21402290
Osman Can Yıldız	21302616

Course Instructor: Prof. Varol Akman

Term Project Report

Dec 27, 2019

This report is submitted to the Department of Computer Engineering Bilkent University in partial fulfillment of the requirements of the Term Project Report course CS461.

1. Introduction

Firing Joel Fagliano is an artificial intelligence-based application, which aims to generate alternative clues to The Mini Puzzle shared by The New York Times. This puzzle is updated daily, according to Greenwich Mean Time (GTM-4), by Joel Fagliano. Today’s puzzle can be found on The New York Times official website (see this link https://www.nytimes.com/crosswords/game/mini) [1].

Firing Joel Fagliano is not an ordinary web crawler that gathers information from the internet and stores in an unfashionable way. It is fully functional artificial intelligence that can generate meaningful clues for any given clue. It uses complex algorithms to decide whether a generated clue is valid or not. Then, it creates a graphical user interface (GUI) to display generated clues with original clues side by side for comparison.

More information about how Firing Joel Fagliano generates valid clues will be explained in the following sections with example screenshots.

2. System Decomposition

Firing Joel Fagliano uses client-server architectural pattern to keep the structure of the program simple. Client subsystem is responsible for displaying puzzle layout, original clues and generated clues. Server subsystem performs the rest of the operation which can be grouped into the following categories in terms of functionality: data retrieval, artificial intelligence and natural language processing. In short, client subsystem is our front-end, and server subsystem is our back-end.

Data retrieval is responsible for gathering information about puzzle and finding possible candidate clues. Data retrieval gathers information in HTML format which is not usable without processing. Thus, it formats and filters gathered information to generate readable results.

Artificial intelligence organizes data retrieval subsystem and decides whether a generated clue is valid or not.

Natural language processing can be considered as subsection of artificial intelligence subsystem. However, it performs highly complicated set of operations on the data which requires well-detailed explanation. We use this feature when data retrieval system fails, and generate a new set of answers by using original solutions to be used for online search.

3. Implementation

3.1. Client

We used React JS to implement our GUI elements. React JS is a JavaScript library used for developing dynamic websites. In other words, React JS can create and update GUI elements independently without loading every element over and over. Moreover, it updates one element at a time to give better understanding of how the program actually works. At the start, browser displays an empty puzzle layout and empty table for clues, but it loads each element one by one [2].

The client sends a REST API (GET) request to the server for each empty place and the server sends the result in JSON format. Then, the client re-formats this data to display on the screen [3].

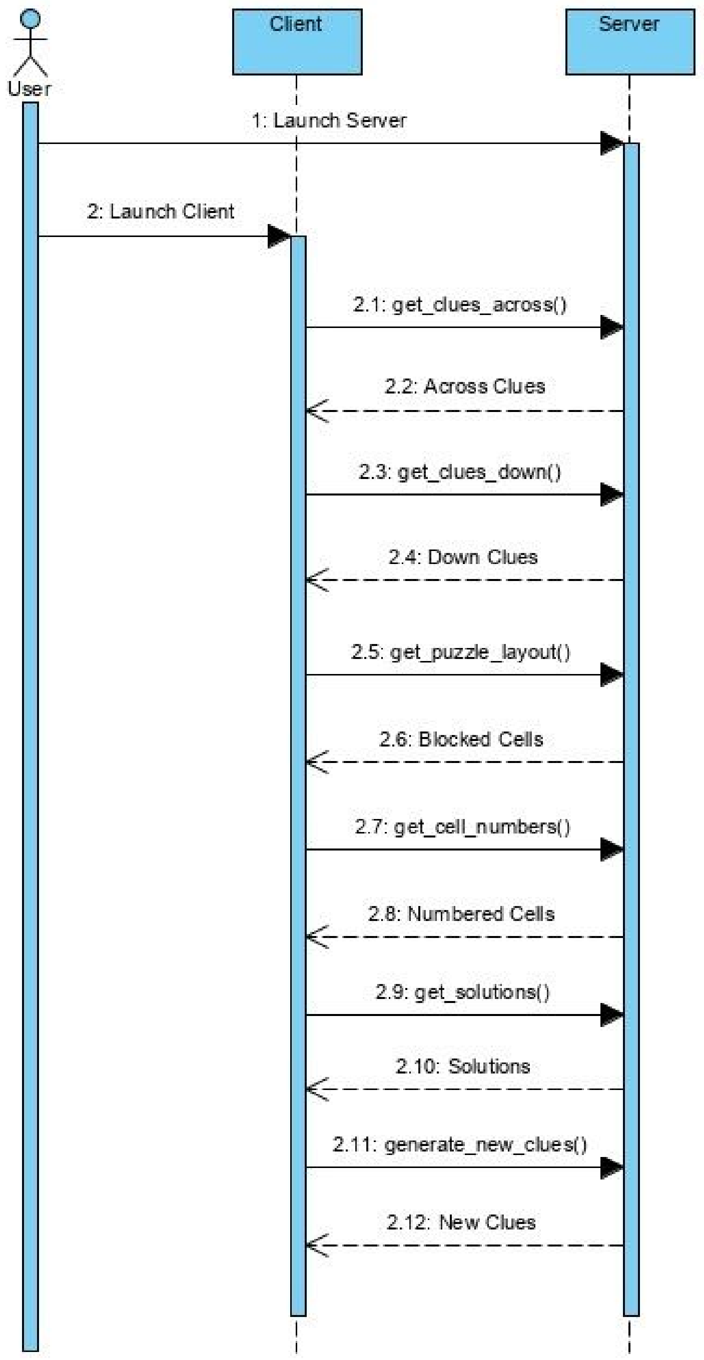
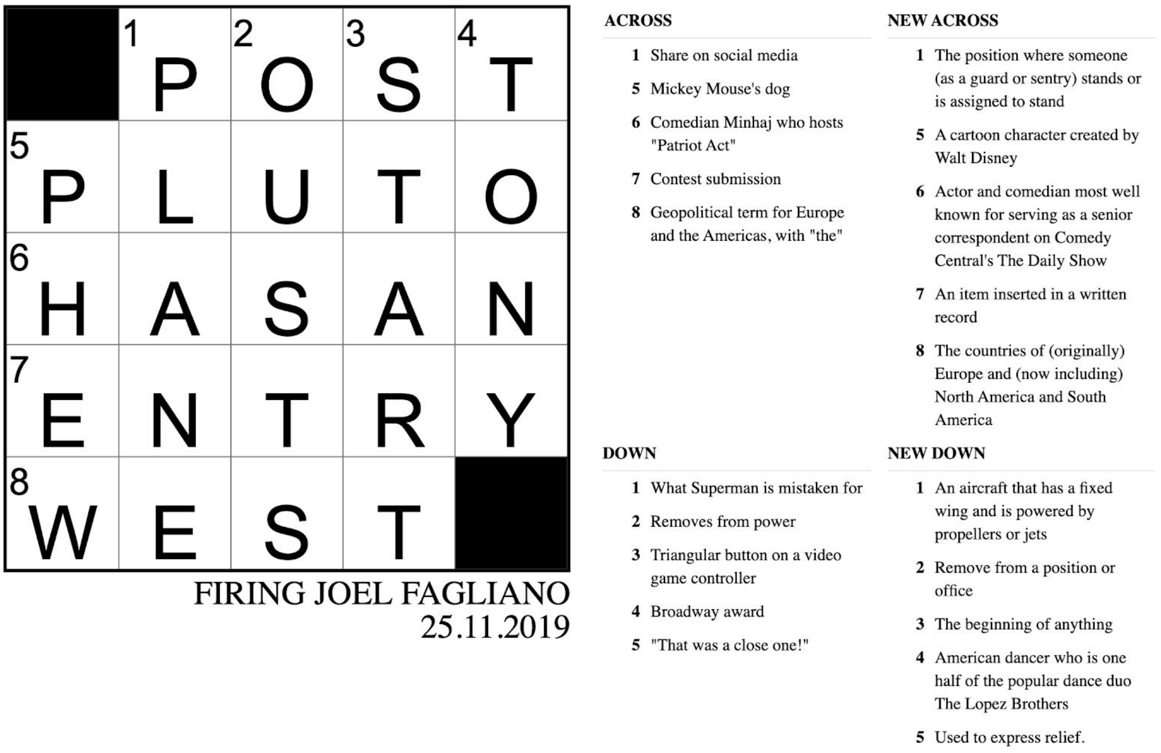


Figure 1: Client-Server Sequence Diagram

Sequence diagram above shows the order of execution between the client and the server. This process takes around 15 to 60 seconds depending on the internet speed and that day's puzzle (some puzzles are harder to solve). The resulting visual can be seen in the figure below.



■	1 P	2 O	3 S	4 T
5 P	L	U	T	O
6 H	A	S	A	N
7 E	N	T	R	Y
8 W	E	S	T	■

FIRING JOEL FAGLIANO
25.11.2019

ACROSS

1 Share on social media

5 Mickey Mouse's dog

6 Comedian Minhaj who hosts "Patriot Act"

7 Contest submission

8 Geopolitical term for Europe and the Americas, with "the"

NEW ACROSS

1 The position where someone (as a guard or sentry) stands or is assigned to stand

5 A cartoon character created by Walt Disney

6 Actor and comedian most well known for serving as a senior correspondent on Comedy Central's The Daily Show

7 An item inserted in a written record

8 The countries of (originally) Europe and (now including) North America and South America

DOWN

1 What Superman is mistaken for

2 Removes from power

3 Triangular button on a video game controller

4 Broadway award

5 "That was a close one!"

NEW DOWN

1 An aircraft that has a fixed wing and is powered by propellers or jets

2 Remove from a position or office

3 The beginning of anything

4 American dancer who is one half of the popular dance duo The Lopez Brothers

5 Used to express relief.

Figure 2: User Interface Example

For more examples, see Appendix...

3.2. Server

We used Python Flask to implement the server. The server can be divided into following three sections in terms of functionality: data retrieval, artificial intelligence and natural language processing [4].

3.2.1. Data Retrieval

Data retrieval is responsible for gathering information about puzzle, finding possible candidate clues and formatting all the information gathered in human readable format. It uses a Mozilla Firefox browser to crawl on the internet. Using a browser allows use to interact with JavaScript elements which The Mini Puzzle website includes [1].

In Figure 1, the client sends multiple requests the server to get information about The Mini Puzzle. This part does not include any decision-making. Therefore, data retrieval can handle this part without following artificial intelligence's instructions. However, the last operation, generate new clues, requires artificial intelligence because it is related to online search and validation of generated clues.

3.2.2. Artificial Intelligence

It was stated that data retrieval is responsible for gathering information, but artificial intelligence is the one that decides the order of search engines. Moreover, it also decides whether gathered information is useful or not.

Unfortunately, this program works only for predetermined search engines. However, we created a large collection of websites just to make sure our program generates a valid clue. Our search engine selection includes following websites: Wordnet, Cambridge, Merriam-Webster, Wikipedia, Urban Dictionary and Famous Birthdays [5, 6, 7, 8, 9, 10].

Before starting the implementation of the project, we decided to solve The Mini Puzzle each day to get a better understanding about the relation between solution and clue. We were able to observe that most of the clues can be generated by an online English dictionary. Wordnet, Cambridge and Merriam-Webster mainly used for this operation [5, 6, 7]. We realized that some solutions had no English meaning, instead they were proper nouns. Proper nouns includes name of places and names of famous people. We used Famous Birthdays to find famous persons, and used Wikipedia for both places and people [8, 10]. We observed that there were some words which had meaning in English, but only used as a slang. Therefore, we added Urban Dictionary. Urban Dictionary is a slang dictionary and provides use information about slang words [9]. Lastly, we realized that some solutions were abbreviations. Although many dictionaries includes meanings for abbreviations in its records, they usually indicate the extended version of the word which is not a valid clue. We implemented an algorithm that can detect if the definition is extended version of abbreviation. Then, we used this definition to search in English dictionaries (Cambridge) to get proper clue.

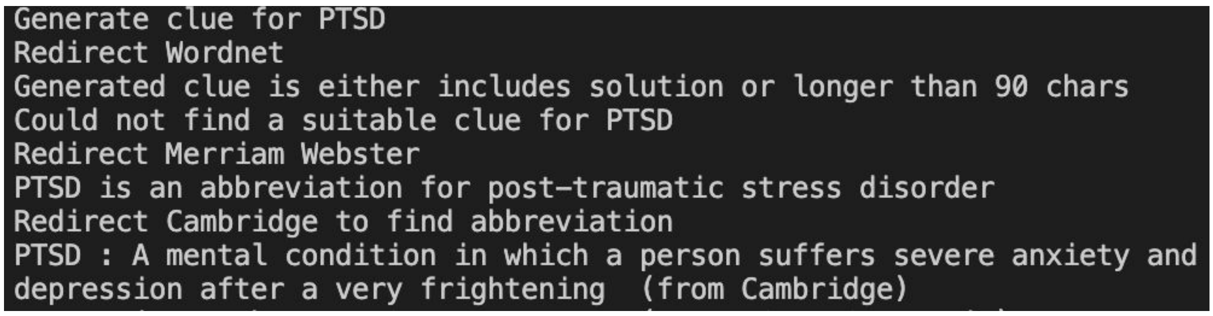

```
Generate clue for PTSD
Redirect Wordnet
Generated clue is either includes solution or longer than 90 chars
Could not find a suitable clue for PTSD
Redirect Merriam Webster
PTSD is an abbreviation for post-traumatic stress disorder
Redirect Cambridge to find abbreviation
PTSD : A mental condition in which a person suffers severe anxiety and
depression after a very frightening  (from Cambridge)
```

Figure 3: Abbreviation Search

Artificial intelligence decides whether clues are valid using a set of rules. These rules vary from search engine to search engine. For some search engines, we found some clever ways to bypass these rules. However, these rules are valid for all the search engines.

Our first rule is that generated clue should not be part of the clue. Artificial intelligence uses regex to validate generated clues. If clue contains its own solution, artificial intelligence marks that clue as valid and continues with the next search engine. However, we figured out a way to convert invalid clues to valid form for Wikipedia. The first sentence in Wikipedia describes the general purpose of the word in the following form: solution + is + definition. Unfortunately, we were not lucky enough to figure out a clever way to solve this problem for other search engines.

Our second rule is that generated clue has to be short. Each search engine has a limit on number of words. It varies from 80 to 120 depending on the search engine. Artificial intelligence checks every clue, and if it is longer than previously initialized character limit, artificial intelligence marks that clue as invalid. Fortunately, we perform a set of operations on each clue to remove unnecessary parts. For example, Wordnet provides descriptions that contains multiple sentences. Although each sentence has some meaning of its own, most people should be able to predict described object using only one of the given sentences. Similar problem occurs in Cambridge dictionary as well. Cambridge provides only one sentence long descriptions. However, some definitions include brackets and an additional definition in between. These additional definitions are also not needed. The goal of dictionary is to provide full knowledge about the subject. However, we realized that this is not the case for crossword puzzles after analyzing The Mini Puzzle.

Our last rule is that extended version of abbreviation cannot be the clue of that abbreviation. Before doing any comparison, generated clue is converted into a word that contains only the first letter of each word. Then, artificial intelligence convert this newly created word with the solution. If they are equal, that means generated clue is extended version of the solution, which is not a good puzzle clue. Fortunately, we find out something interesting, while analyzing Cambridge dictionary. When you search for PTSD in Cambridge dictionary, it gives the following output: post-traumatic stress disorder. However, if you search for post-traumatic stress disorder in Cambridge dictionary, it gives the following output: A mental condition in which a person suffers from severe anxiety and depression after a very frightening or shocking experience, such as an accident or a war. We realized that extended version of the abbreviation can be used for generating a valid clue.

In addition to clue validation, artificial intelligence is responsible for the order of search engines. We did not initialize a predetermined order for these search engines, instead artificial intelligence dynamically decides the order of search engines. It tries to mimic a tree like structure. We actually did not implement a tree, but our algorithms uses tree logic to choose search engine. Each search engine has a score which is stored in a text file, and updates this text file after each operation depending on the result. Artificial intelligence chooses the search engine with the highest score, and then follows the order of scores until it finds a valid clue. If generated clue is valid, it updates the score of the website which that clue comes from.

All the things mentioned above is related to one word solutions, but there are also solutions that are a combination of multiple words. These solutions only occurs once or twice a week. However, we cannot generate a valid clue for them without performing segmentation on the solution. Natural language processing performs this operation. It uses a probabilistic approach to calculate to find possible word segmentations for that particular word, and it returns all the possible segmentations. Artificial intelligence uses these segmentations in given order to generate a valid clue. More information about how natural language processing calculates mentioned probabilities will be explained under its own section.

3.2.3. Natural Language Processing

In the previous section, it was stated that natural language processing is responsible for finding possible word segmentations in the best order. The program does that by calculating the probability of each split happening. For example, the word "ALLIN" (a real example from The Mini Puzzle) can be divided in four different ways as follows: "A LLIN", "AL LIN", "ALL IN", "ALLI N". For each split, the program tries to calculate the probability of each word in that split, and then calculates the conditional probability of the second word coming after the first one.

This operation can be described as follows using mathematical symbols.

$$\text{P(Second Word | First Word)} = \frac{\text{P(First Word + ' ' + Second Word)}}{\text{P(First Word)}}$$

- For "ALL IN", calculations as follows.

$$\text{P("IN" | "ALL")} = \frac{\text{P("ALL IN")}}{\text{P("ALL")}} = \frac{0.0003498542274052478}{0.004193040087599818} = 0.08343689068$$

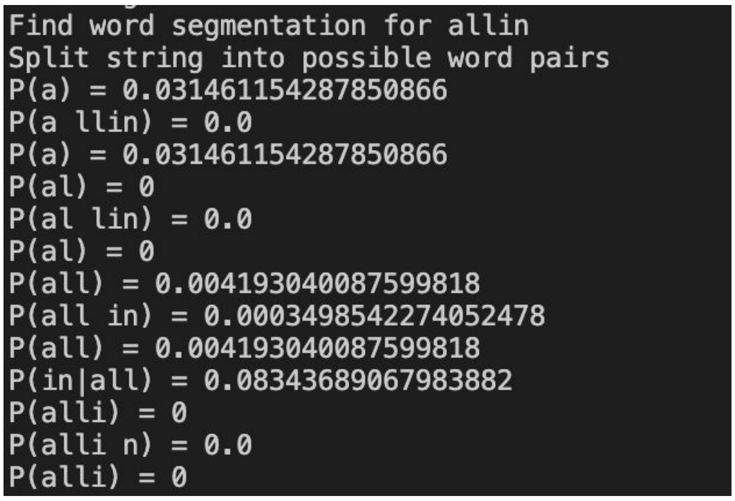
In other words, "IN" coming after "ALL" has probability of 0.08343689068.

- For "A LLIN", calculations as follows.

$$\text{P("LLIN" | "A")} = \frac{\text{P("A LLIN")}}{\text{P("A")}} = \frac{0}{0.031461154287850866} = 0$$

Although "A" means something in English, "A LLIN" has no meaning. Thus, their probability becomes 0.

- For "AL LIN" and "ALLI N", we assume that the conditional probability is 0 because division operation raises division by zero error.



```
Find word segmentation for allin
Split string into possible word pairs
P(a) = 0.031461154287850866
P(a llin) = 0.0
P(a) = 0.031461154287850866
P(al) = 0
P(al lin) = 0.0
P(al) = 0
P(all) = 0.004193040087599818
P(all in) = 0.0003498542274052478
P(all) = 0.004193040087599818
P(in|all) = 0.08343689067983882
P(alli) = 0
P(alli n) = 0.0
P(alli) = 0
```

Figure 4: Probability Distribution for "ALLIN"

Finding conditional probability is a simple but effective method. However, this method requires probabilities which were calculated prior to execution of the clue generator.

We calculated these probabilities using a bag of words model. This model can only be created using a book from English literature. There are two important things about this chosen book. It has to have either Free License or Academic Free License, and it must be written in Modern English (a term called Shakespeare English exists which indicates the old tongue that uses words like thou, thee and many other currently unused words).

We decided to use "The Adventures of Sherlock Holmes" by Arthur Conan Doyle. It is Free License, and it does not contain previously mentioned words. Bag of words model only care about the counts of words. We constructed two different models. The first one is called unigram (1-gram). This model considered each word as separately element. The second model is called bigram (2-gram). It assumed that sequence of two adjacent words as one element. The aim of both models is to count the number of times an element occurred in that particular text. For both models, we added an extra filter which allowed only elements that are shorter or equal to five characters (The Mini Puzzle is 5 by 5).

For example, we have the following text. “It was not the wife; it was the children, groaned the prisoner.” (a random line from The Adventures of Sherlock Holmes). For this text, our models will give use the following outputs.

Unigram Bag of Words = {"it" : 2, "was" : 2, "not" : 1, "the" : 3, "wife" : 1}

Bigram Bag of Words = {"it was" : 2}

Then, we can calculate the probability of each word, and write in a text file.

4. Conclusion

To conclude, Firing Joel Fagliano is built to generate valid clues from The New York Times’ daily The Mini Puzzle. In order to achieve that we crawled the solutions using a web crawler. Then, we used another crawler to find possible candidate clues. Many complex artificial algorithms performed on these clues to find whether they are valid. These algorithms include tree structure, abbreviation comparison, regex, size limitation and word segmentation. Lastly, we created a nice looking graphical user interface to display our results.

This project reports work done in partial fulfillment of the requirements for CS 461 -- Artificial Intelligence. The software is, to a large extent, original (with borrowed code clearly identified) and was written solely by members of the project group FIRING JOEL FAGLIANO.

Word Count (excluding headers, figure indicators, mathematical equations, citations and figures) : 2,136

References

[1] “The New York Times Mini Crossword,” *The New York Times,* The New York Times, https://www.nytimes.com/crosswords/game/mini

[2] “React - A JavaScript library for building user interfaces,” *A JavaScript library for building user interfaces,* https://reactjs.org.

[3] “Representational state transfer,” *Wikipedia,* https://en.m.wikipedia.org/wiki/Representational_state_transfer.

[4] “Flask,” *Full Stack Python*, https://www.fullstackpython.com/flask.html.

[5] “What is WordNet?,” *Princeton University*, https://wordnet.princeton.edu.

[6] “Dictionary by Merriam-Webster: America's most-trusted online dictionary,” *Merriam-Webster,* https://www.merriam-webster.com.

[7] “Cambridge English Dictionary: Meanings & Definitions,” *Cambridge Dictionary*, https://dictionary.cambridge.org/dictionary/english/.

[8] “Main Page,” *Wikipedia*, https://en.wikipedia.org/wiki/Main_Page.

[9] “Urban Dictionary,” Urban Dictionary, https://www.urbandictionary.com.

[10] “Famous Birthdays,” Famous Birthdays, https://www.famousbirthdays.com

Appendix A: Codes

A1: Preprocess

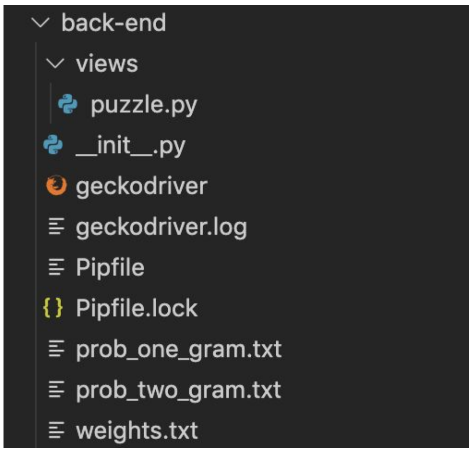
Run create_prob_dist.py prior to anything else. It should generate text files that includes probabilities, and then copy them to back-end folder.

create_prob_dist.py

```
import requests
import re
from bs4 import BeautifulSoup
from nltk.util import ngrams
# Title : The Adventures of Sherlock Holmes, by Arthur Conan Doyle
# This book is free to use. Therefore, it will not cause any legal trouble (Free
license)
link = 'https://www.gutenberg.org/files/1661/1661-h/1661-h.htm'
html = requests.get(link).text
b_soup = BeautifulSoup(html, 'html.parser')

# Create a text file to write
text_file_1 = open("prob_one_gram.txt","w+")
text_file_2 = open("prob_two_gram.txt","w+")

# This link includes some information about the provider, but we are only looking for
what is written in the book
# Therefore, we are extracting unnecessary stuff
text = ''
for chapters in b_soup.find_all('div', {'class' : 'chapter'}):
 text = text + chapters.text

# 1-Gram
# Tokenize each word to count later
# [a-z][a-z]?[a-z]?[a-z]?[a-z]? = Word with at most 5 letters surrounded by spaces
words_1 = re.findall(' [a-z][a-z]?[a-z]?[a-z]?[a-z]? ', text.lower())
words_1 = [word[1:-1] for word in words_1] # Remove empty spaces around tokens

# Calculate frequency of each word
freqs_1 = [words_1.count(x) for x in words_1]
freq_dict_1 = dict(list(zip(words_1, freqs_1)))
sorted_freq_list_1 = sorted(freq_dict_1.items(), reverse=True, key = lambda x: x[1])

# Create a probability distribution using frequency
sorted_freq_list_1 = [[word, freq / len(words_1)] for (word, freq) in
sorted_freq_list_1]

# Write and close text file
for word in sorted_freq_list_1:
 text_file_1.write(str(word[0]) + " " + str(word[1]) + '\n')
text_file_1.close()

# 2-Grams
# Tokenize each two words to count later
# A phrase can be at most 5 char (excluding space)
words_2 = re.findall(' [a-z] [a-z] ', text.lower()) # Case 1 : " _ _ " (_ indicates
char)
words_2.extend(re.findall(' [a-z][a-z] [a-z] ', text.lower())) # Case 2 : "__ _"
words_2.extend(re.findall(' [a-z] [a-z][a-z] ', text.lower())) # Case 3 : "_ __"
words_2.extend(re.findall(' [a-z][a-z] [a-z][a-z] ', text.lower())) # Case 4 : "__ __"
words_2.extend(re.findall(' [a-z][a-z][a-z] [a-z][a-z] ', text.lower())) # Case 5 : "___
```

```
__"
words_2.extend(re.findall(' [a-z][a-z] [a-z][a-z][a-z] ', text.lower())) # Case 6 : "__
___"
words_2 = [phrase[1:-1] for phrase in words_2] # Remove empty spaces around tokens

# Calculate frequency of each word
freqs_2 = [words_2.count(x) for x in words_2]
freq_dict_2 = dict(list(zip(words_2, freqs_2)))
sorted_freq_list_2 = sorted(freq_dict_2.items(), reverse=True, key = lambda x: x[1])

# Create a probability distribution using frequency
sorted_freq_list_2 = [[word, freq / len(words_2)] for (word, freq) in
sorted_freq_list_2]

# Write and close text file
for word in sorted_freq_list_2:
 text_file_2.write(str(word[0]) + " " + str(word[1]) + '\n')
text_file_2.close()
```

A2: Back-end



```
back-end
  views
    puzzle.py
  __init__.py
  geckodriver
  geckodriver.log
  Pipfile
  Pipfile.lock
  prob_one_gram.txt
  prob_two_gram.txt
  weights.txt
```

Create a pip environment and install packages listed in Pipfile.

Download a geckodriver for Mozilla Firefox from official github page (see this link https://github.com/mozilla/geckodriver).

Create a text file named weights.txt and enter given values.

Type the following commands in back-end directory.

$ pipenv shell

$ export FLASK_DEBUG=0

$ export FLASK_APP=./

$ flask run

__init__.py

```
from flask import Flask
from flask_cors import CORS

def create_app():
  app = Flask(__name__)
  CORS(app)

  from .views.puzzle import puzzle
  app.register_blueprint(puzzle)

  return app

# export FLASK_DEBUG=0
# export FLASK_APP=./
```

puzzle.py

```
from flask import Blueprint, jsonify
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from bs4 import BeautifulSoup
import re
import math

# Init Blueprint
puzzle = Blueprint('puzzle', __name__)

# Init Selenium
def open_browser():
  try:
    global driver
    driver = webdriver.Firefox(executable_path="/Users/cagataykupeli/Desktop/CS
461/Project/Puzzle/back-end/geckodriver")
  except:
    pass

# Helper Functions
def redirect_puzzle_page():
  try:
    driver.get("https://www.nytimes.com/crosswords/game/mini")
```

```
   #driver.get("https://www.nytimes.com/crosswords/game/mini/2016/06/01") # Use it
during Saturdays for testing
 except:
   pass


def inspect_old_clues():
 old_clues_across = {} # Empty dictionary
 old_clues_down = {} # Empty dictionary
 b_soup = BeautifulSoup(driver.page_source, 'html.parser')
 print('Get old clues')
 for i, (old_clue_label, old_clue) in enumerate(zip(b_soup.find_all('span', {'class' :
'Clue-label--2IdMY'}), b_soup.find_all('span', {'class' : 'Clue-text--3lZl7'}))):
   if(i < 5):
     old_clues_across[old_clue_label.text] = old_clue.text
   else:
     old_clues_down[old_clue_label.text] = old_clue.text
 return old_clues_across, old_clues_down

def inspect_puzzle_layout():
 print('Get blocked cell locations')
 blocked_cells = []
 b_soup = BeautifulSoup(driver.page_source, 'html.parser')
 for blocked_cell in b_soup.find_all('rect', {'class' : 'Cell-block--1oNaD'}):
   blocked_cells.append(int(blocked_cell.get('id')[8:]))
 return blocked_cells

def inspect_cell_numbers():
 print('Get cell numbers')
 cell_numbers = []
 b_soup = BeautifulSoup(driver.page_source, 'html.parser')
 for cell_text_anchor in b_soup.find_all('text', {'text-anchor' : 'start'}):
   cell_number = cell_text_anchor.find_previous_sibling('rect').get('id')[8:]
   cell_numbers.append(int(cell_number))
 return cell_numbers

def reveal_solutions():
 print('Reveal solutions')
 try:
   WebDriverWait(driver, 5).until(
     EC.presence_of_element_located((By.XPATH,
'/html/body/div[1]/div/div/div[4]/div/main/div[2]/div/div[2]/div[3]/div/article/div[2]/b
utton'))
   ).click()
 except:
   pass
 try:
   WebDriverWait(driver, 5).until(
     EC.presence_of_element_located((By.XPATH,
'/html/body/div[1]/div/div/div[4]/div/main/div[2]/div/div/ul/div[2]/div[1]/li/button'))
   ).click()
 except:
   pass
 try:
   WebDriverWait(driver, 5).until(
     EC.presence_of_element_located((By.XPATH,
'/html/body/div[1]/div/div/div[4]/div/main/div[2]/div/div/ul/div[2]/li[2]/button'))
   ).click()
 except:
   pass
 try:
   WebDriverWait(driver, 5).until(
     EC.presence_of_element_located((By.XPATH,
'/html/body/div[1]/div/div/div[4]/div/main/div[2]/div/div/ul/div[2]/li[2]/ul/li[3]/a'))
   ).click()
 except:
   pass
 try:
   WebDriverWait(driver, 5).until(
     EC.presence_of_element_located((By.XPATH,
'/html/body/div[1]/div/div[2]/div[2]/article/div[2]/button[2]'))
   ).click()
 except:
   pass
 try:
   WebDriverWait(driver, 5).until(
     EC.presence_of_element_located((By.XPATH,
'/html/body/div[1]/div/div[2]/div[2]/span'))
   ).click()
 except:
   pass

def inspect_solutions():
 print('Get solutions')
 solutions = []
 for _ in range(0, 25):
   solutions.append('')
 b_soup = BeautifulSoup(driver.page_source, 'html.parser')
 for letter in b_soup.find_all('text', {'text-anchor' : 'middle'}):
   # For some reason at 26, Nov, NY Times changed their html structure. As a result,
this monstrosity occured. I could just use the script in if statement. However, I fear
that they might change the structure again.
   if(len(letter.text) > 1):
     solutions[int(letter.find_previous_sibling('rect').get('id')[8:])] = letter.text[0]
   elif(len(letter.text) == 1):
     solutions[int(letter.find_previous_sibling('rect').get('id')[8:])] = letter.text
 return solutions

def convert_solutions(solution_list):
 print('Get solutions')
 solutions_dict = {}
 b_soup = BeautifulSoup(driver.page_source, 'html.parser')
 for i in b_soup.find_all('text', {'text-anchor' : 'start'}):
   index = int(i.find_previous_sibling('rect').get('id')[8:])
   # Down
   if(index < 5):
     solutions_dict[i.text + 'D'] = ''.join([solution_list[index], solution_list[5 +
index], solution_list[10 + index], solution_list[15 + index], solution_list[20 +
index]])
   elif((index >= 5) & (solution_list[index - 5] == '')):
     solutions_dict[i.text + 'D'] = ''.join([solution_list[index % 5], solution_list[5 +
(index % 5)], solution_list[10 + (index % 5)], solution_list[15 + (index % 5)],
solution_list[20 + (index % 5)]])
   # Across
   if((index % 5 != 0) & (solution_list[index - 1] == '')):
     solutions_dict[i.text + 'A'] = ''.join(solution_list[(index // 5) * 5:((index // 5)
* 5 + 5)])
   elif((index % 5 == 0) & (solution_list[index] != '')):
     solutions_dict[i.text + 'A'] = ''.join(solution_list[(index // 5) * 5:((index // 5)
* 5 + 5)])
 return solutions_dict

def search_famous_person(solution):
 driver.get('https://www.famousbirthdays.com/names/' + solution.lower() + '.html')
 b_soup = BeautifulSoup(driver.page_source, 'html.parser')
 if(b_soup.find(text = 'Oops!') == None):
   href = b_soup.find('a', {'class' : 'face person-item'}).get('href')
   driver.get(href)
   b_soup = BeautifulSoup(driver.page_source, 'html.parser')
   [x.replace_with(x.text) for x in b_soup.find_all('a')]
   # Replace nonbreak space with regular space
   new_clue = b_soup.find('p').text[:-1].replace('\xa0', ' ') # \xa0 is unicode of
nonbreak space
   # If clue is more than 2 sentences, use only the 1st one
   for i, l in enumerate(new_clue):
     if(l == '.'):
       new_clue = new_clue[:i]
   # Clue supposed to be 1 sentence long. However, it is still longer than 120 chars
   if(len(new_clue) > 120):
     end_sentence = re.search('" [A-Z]', new_clue)
     if(end_sentence != None):
       end_index = re.search('" [A-Z]', new_clue).end()
       new_clue = new_clue[:end_index - 2]
   # If the word is mentioned in clue, return nothing
   '''
   if(new_clue.lower().find(solution.lower()) == -1):
     return new_clue
   else:
     return ''
   '''
   return new_clue
 else:
   print('No famous person called ' + solution.upper() + ' exists')
   return ''

def search_wordnet(solution):
 driver.get('http://wordnetweb.princeton.edu/perl/webwn?s=' + solution.lower())
 b_soup = BeautifulSoup(driver.page_source, 'html.parser')
 [x.extract() for x in b_soup.find_all('a')]
 [x.extract() for x in b_soup.find_all('b')]
 [x.extract() for x in b_soup.find_all('i')]
 html = b_soup.find('li')
 if(html != None):
   new_clue = html.text
   new_clue = new_clue[:-2]
   for i, l in enumerate(new_clue):
     if(l == '('):
       new_clue = new_clue[i + 1:]
       break
   new_clue_list = new_clue.split(';')
   #print(new_clue_list)
   for i, clue in enumerate(new_clue_list):
     # If word is not the 1st clue in the list, it starts with ' '. Thus, we eliminate
that by removing that ' '.
     if(i != 0):
       clue = clue[1:]
     if((clue.lower().find(solution.lower()) == -1) & (len(clue) < 90)):
       # Check if it is an abbreviation
       extended_abb = clue
       clue = clue.replace('-', ' ') # PTSD might be written as post-traumatic stress
disorder
       abb_list = clue.split(' ')
       abb_checker = ''
       for abb in abb_list:
         abb_checker = abb_checker + abb[0]
       if(abb_checker.lower() != solution.lower()):
         clue = clue[0].upper() + clue[1:]
         return clue
       else:
         print(solution.upper() + ' is an abbreviation for ' + extended_abb)
         return find_abbreviation_meaning(extended_abb)
     else:
       print('Generated clue is either includes solution or longer than 90 chars')
   print('Could not find a suitable clue for ' + solution.upper())
   return ''
 else:
   print('Wordnet has no record for ' + solution.upper())
   return ''

def search_urban(solution):
 driver.get('https://www.urbandictionary.com/define.php?term=' + solution.lower())
 b_soup = BeautifulSoup(driver.page_source, 'html.parser')
 if(b_soup.find('div', {'class' : 'shrug space'}) == None):
   new_clue = b_soup.find('div', {'class' : 'meaning'}).text
   # Remove '.' at the end if exists
   if(new_clue[-1] == '.'):
     new_clue = new_clue[:-1]
   # If the length of clue is more than 120 (4 Lines) chars, return nothing
   if(len(new_clue) > 120):
     print('Generated clue is longer than 120 chars')
     return ''
   else:
     new_clue = new_clue[0].upper() + new_clue[1:]
     # If a word is mentioned in clue, return nothing
     if(new_clue.lower().find(solution.lower()) == -1):
       return new_clue
```

```
    else:
      print('Generated clue includes ' + solution.upper())
      return ''
 else:
   print('Urban Dictionary has no record for ' + solution.upper())
   return ''

def search_merriam_webster(solution):
 driver.get('https://www.merriam-webster.com/dictionary/' + solution.lower())
 b_soup = BeautifulSoup(driver.page_source, 'html.parser')
 # There is two ways this site represents its data
 if(b_soup.find('h1', {'class' : 'mispelled-word'}) == None):
   if(b_soup.find('span', {'class' : 'dtText'}) != None):
     new_clue = b_soup.find('span', {'class' : 'dtText'}).text
     new_clue = new_clue.replace(':', '') # For some reason, some descriptions starts
with ':'
     extended_abb = new_clue # Store this information just in case if found solution is
abb
     # If word starts with ':', it continues with a ' '
     new_clue = new_clue.replace('-', ' ')
     # Merriam webster is a strange website. There are so many ways to display
definition.
     # One includes example as well, following section aims to deal with that particular
case
     for i, l in enumerate(new_clue):
       if(l == '\n'):
         new_clue = new_clue[:i - 1]
         break
     while(new_clue[-1] == ' '):
       new_clue = new_clue[:-1]
     if(new_clue[0] == ' '):
       new_clue = new_clue[1:]
     # Check if it is an abbreviation
     abb_list = new_clue.split(' ')
     abb_checker = ''
     for abb in abb_list:
       abb_checker = abb_checker + abb[0]
     if(abb_checker.lower() != solution.lower()):
       new_clue = new_clue[0].upper() + new_clue[1:]
       if(len(new_clue) < 120):
         return new_clue, 'Merriam Webster'
       else:
         print('Generated clue is longer than 80 chars')
         return '', 'Merriam Webster'
     else:
       print(solution.upper() + ' is an abbreviation for ' + extended_abb)
       return find_abbreviation_meaning(extended_abb), 'Cambridge'
   else:
     if(b_soup.find('span', {'class' : 'mw_t_sp'}) != None): # If the word is an abb,
check for an example sentence
       new_clue = b_soup.find('span', {'class' : 'mw_t_sp'}).text
       new_clue = new_clue.lower()
       placeholder = ''.join(['_' for _ in range(0, len(solution))])
       new_clue = new_clue.replace(solution.lower(), placeholder)
```

```
        new_clue = new_clue[0].upper() + new_clue[1:]
        new_clue = '"' + new_clue + '"'
        if(new_clue[-1] == '.'):
          new_clue = new_clue[:-1]
        if(len(new_clue) < 120):
          return new_clue, 'Merriam Webster'
        else:
          print('Generated clue is longer than 80 chars')
          return '', 'Merriam Webster'
      else:
        if(b_soup.find('span', {'class' : 'unText'}) != None):
          new_clue = b_soup.find('span', {'class' : 'unText'}).text
          new_clue = new_clue[0].upper() + new_clue[1:]
          return new_clue
    print('Could not find a suitable clue for ' + solution.upper())
    return '', 'Merriam Webster'
  else:
    print('Merriam Webster has no record for ' + solution.upper())
    return '', 'Merriam Webster'

def search_cambridge(solution): # Default value of spacing is false
  driver.get('https://dictionary.cambridge.org/dictionary/english/' + solution)
  b_soup = BeautifulSoup(driver.page_source, 'html.parser')
  new_clue_obj = b_soup.find('div', {'class' : 'def ddef_d db'})
  if(new_clue_obj != None):
    new_clue = new_clue_obj.text
    # Some of the abbreviation states that it is and abbreviation at the start of
sentence
    # If abbreviation word stated, it is in the form of "abbreviation for sth
(=definition)"
    if(re.search('abbreviation', new_clue) != None):
      for i, l in enumerate(new_clue):
        if(l == '='):
          if(new_clue[i - 1] == '('):
            new_clue = new_clue[i + 2:-1]
    else:
      # If no abbreviation word appears, it sometimes still have the form "definition 1
(=definition 2)",
      # but definition 1 is always better
      for i, l in enumerate(new_clue):
        if(l == '='): # If it has the form definition 1 only. It will not enter this if
statement and use definition 1 only
          if(new_clue[i - 1] == '('):
            new_clue = new_clue[:i - 1]
    # Remove parts after or
    if(re.search(' or ', new_clue) != None):
      for i, l in enumerate(new_clue):
        if((new_clue[i] == 'o') and (new_clue[i + 1] == 'r')):
          new_clue = new_clue[:i]
          break
    # If the word is a short form of another word, it has the form "short form of long
form"
    # However, they usually provide an example
    new_clue = new_clue.replace(':', '')
```

```python
  if(re.search('short form', new_clue) != None):
   new_clue = b_soup.find('div', {'class' : 'examp dexamp'}).text
   new_clue = new_clue.lower()
   new_clue = new_clue.replace(solution.lower(), '_____')
  new_clue = new_clue[0].upper() + new_clue[1:]
  if(len(new_clue) < 120):
   return new_clue
  else:
   print('Generated clue is longer than 120 chars')
   return ''
 else:
  print('Cambridge has no record for ' + solution.upper())
  return ''

def search_wikipedia(solution):
 driver.get('https://en.wikipedia.org/wiki/' + solution)
 b_soup = BeautifulSoup(driver.page_source, 'html.parser')
 if(b_soup.find('div', {'class' : 'noarticletext mw-content-ltr'}) == None):
  [x.extract() for x in b_soup.find_all('div', {'role' : 'note'})]
  [x.extract() for x in b_soup.find_all('table', {'role' : 'presentation'})]
  new_clue = ''
  # Temporary state
  temp = b_soup.find('div', {'class' : 'mw-parser-output'})
  if(re.search('may refer to', temp.p.text) == None):
   new_clue = temp.find('p').text
   for i, l in enumerate(new_clue):
    if(l == '.'):
     new_clue = new_clue[:i]
   exists_is = re.search('is ', new_clue)
   if(exists_is != None):
    end_index = exists_is.end()
    new_clue = new_clue[end_index:]
    new_clue = new_clue[0].upper() + new_clue[1:]
    if(len(new_clue) < 120):
     return new_clue
    else:
     print('Generated clue is longer than 120 chars')
     return ''
   else:
    print('Generated clue is formatted incorrectly')
    return ''
  else:
   new_url = temp.find('li').find('a').get('href')
   if(new_url == None):
    print('Generated clue is formatted incorrectly')
    return new_clue
   else:
    print('More than one wikipedia page for ' + solution.upper())
    print('Redirect first page on the list')
    driver.get('https://en.wikipedia.org' + new_url)
    b_soup_new = BeautifulSoup(driver.page_source, 'html.parser')
    new_clue = b_soup_new.find('div', {'class' : 'mw-parser-output'}).find('p').text
    for i, l in enumerate(new_clue):
     if(l == '.'):
       new_clue = new_clue[:i]
    exists_is = re.search('is ', new_clue)
    if(exists_is != None):
     end_index = exists_is.end()
     new_clue = new_clue[end_index:]
     new_clue = new_clue[0].upper() + new_clue[1:]
     if(len(new_clue) < 120):
       return new_clue
     else:
       print('Generated clue is longer than 120 chars')
       return ''
    else:
     print('Generated clue is formatted incorrectly')
     return ''
   return new_clue
 else:
  print('Wikipedia has no record for ' + solution.upper())
  return ''

# Takes an extended version of abbreviation and tries to find a suitable solution
# It is used when algorithm detects possibility of abbreviation
# For example, when you search PTSD, it gives post-traumatic stress disorder. We know
that this is extended version of PTSD because
# Post
# Traumatic
# Stress
# Disorder
def find_abbreviation_meaning(expanded_solution):
 print('Redirect Cambridge to find abbreviation')
 # Use cambridge for finding abbreviation meaning
 expanded_solution = expanded_solution.lower()
 return search_cambridge(expanded_solution)

# Split a string into possible word pairs
def split_string(string):
 print('Split string into possible word pairs')
 return [(string[:i], string[i:]) for i in range(1, min(len(string), 5))] # Excluding
the word itself

# Return the probability of one word if it exists in our text file
# P(A)
def prob_1(word):
 prob = 0
 with open('./prob_one_gram.txt','r') as text_file:
  for line in text_file:
   line_list = line.split()
   if(line_list[0] == word):
    prob = line_list[1]
 print('P(' + word + ') =', prob)
 return float(prob)

# Returns the probability of two word phrases if it exists in our text file
# P(A, B)
def prob_2(words):
 prob = 0
 with open('./prob_two_gram.txt','r') as text_file:
   for line in text_file:
     line_list = line.split()
     if(line_list[0] + ' ' + line_list[1] == words):
       prob = line_list[2]
 print('P(' + words + ') =', float(prob))
 return float(prob)

# Returns the probability of two words phrases given previous word
# P(A | B) = P(A, B) / P(B)
def cond_prob(word, prev_word):
 prob_union = prob_2(prev_word + ' ' + word) # P(word, prev_word)
 prob_prev = prob_1(prev_word) # P(prev_word)
 if((prob_union > 0) & (prob_prev > 0)):
   cond_prob = prob_union / prob_prev
   print('P(' + word + '|' + prev_word + ') =', cond_prob)
   return cond_prob
 else:
   # Probability cannot be less than 0
   # If phrase (word + ' ' + word) does not exist, there is no point of calculating
probability
   # If previous word does not exist, there is no point of calculating probability
   # (If previous word is missing, there should not be a combination)
   return 0

# Both find_best_segmentation_ind and find_best_segmentation_cond requires a word such
as "weare" not "we are". Thus, directly insert gathered solution without splitting
# Find the most possible word pair for independent words (Unused)
def find_best_segmentation_ind(string):
 candidates = split_string(string)
 probs = []
 for first, second in candidates:
   prob_first = prob_1(first)
   prob_second = prob_1(second)
   if((prob_first == 0) & (prob_second == 0)):
     probs.append(0)
   elif((prob_first != 0) & (prob_second != 0)):
     probs.append(prob_first * prob_second)
   elif((prob_first != 0) & (prob_second == 0) & (len(second) == 0)):
     probs.append(prob_first)
   elif((prob_first == 0) & (prob_second != 0) & (len(first) == 0)):
     probs.append(prob_first)
   else:
     probs.append(0)
 sorted_candidates = [x for _, x in sorted(zip(probs, candidates))]
 sorted_candidates.reverse()
 return sorted_candidates

# Find the most possible word pair using conditional probability
def find_best_segmentation_cond(string):
 print('Find word segmentation for ' + string)
 candidates = split_string(string)
 probs = []
 for first, second in candidates:
  prob_first = prob_1(first)
  prob_second = cond_prob(second, first)
  if((prob_first == 0) & (prob_second == 0)):
   probs.append(0)
  elif((prob_first != 0) & (prob_second != 0)):
   probs.append(prob_first * prob_second)
  elif((prob_first != 0) & (prob_second == 0) & (len(second) == 0)):
   probs.append(prob_first)
  elif((prob_first == 0) & (prob_second != 0) & (len(first) == 0)):
   probs.append(prob_first)
  else:
   probs.append(0)
 sorted_candidates = [x for _, x in sorted(zip(probs, candidates))]
 sorted_candidates.reverse()
 return sorted_candidates

#   print(solution + ' does not exists in Wikipedia')
def find_alt_clue(solution):
 print('Generate clue for ' + solution.upper())
 solution = solution.lower()
 last_visited = ''
 try:
  new_clue = ''
  weights = {}
  with open('./weights.txt','r') as text_file:
   for line in text_file:
    line_list = line.split()
    weights[line_list[0]] = line_list[1]
    if(new_clue == ''):
     if(line_list[0] == 'wordnet'):
      print('Redirect Wordnet')
      new_clue = search_wordnet(solution)
      last_visited = 'Wordnet'
     elif(line_list[0] == 'cambridge'):
      print('Redirect Cambridge')
      new_clue = search_cambridge(solution)
      last_visited = 'Cambridge'
     elif(line_list[0] == 'merriam'):
      print('Redirect Merriam Webster')
      new_clue, last_visited = search_merriam_webster(solution)
     elif(line_list[0] == 'famous'):
      # Search a famous person
      print('Redirect Famous Person')
      new_clue = search_famous_person(solution)
      last_visited = 'Famous Birthdays'
     elif(line_list[0] == 'wikipedia'):
      print('Redirect Wikipedia')
      new_clue = search_wikipedia(solution)
      last_visited = 'Wikipedia'
     elif(line_list[0] == 'urban'):
      print('Redirect Urban')
      new_clue = search_urban(solution)
      last_visited = 'Urban'
    if(new_clue == ''):
      solution_segmentation_list = find_best_segmentation_cond(solution)
      for segmentation in solution_segmentation_list:
        print('Generate clue for ' + segmentation)
        # Check meriam webster
        if(new_clue == ''):
          print('Redirect Merriam Webster for ' + " ".join(segmentation))
          new_solution = ' '.join(segmentation)
          new_clue, last_visited = search_merriam_webster(new_solution)
          if(new_clue == ''):
            print('Redirect Merriam Webster for ' + "'".join(segmentation))
            new_solution = '%27'.join(segmentation) # Depends on the site's structure
            new_clue, last_visited = search_merriam_webster(new_solution)
            if(new_clue == ''):
              print('Redirect Cambridge for ' + ' '.join(segmentation))
              new_solution = '+'.join(segmentation)
              new_solution = '?q=' + new_solution
              new_clue = search_cambridge(new_solution)
              last_visited = 'Cambridge'
              if(new_clue == ''):
                print('Redirect Cambridge for ' + "'".join(segmentation))
                new_solution = '-'.join(segmentation)
                new_solution = new_solution + '?q='
                new_solution = new_solution + '%27'.join(segmentation)
                new_clue = search_cambridge(new_solution)
                last_visited = 'Cambridge'
  if(new_clue == ''):
    print('Could not generate a clue for ' + solution.upper())
  else:
    print(solution.upper() + ' : ' + new_clue + ' (from ' + last_visited + ')')
    if(last_visited == 'Wordnet'):
      temp = weights['wordnet']
      weights['wordnet'] = int(temp) + 1
    elif(last_visited == 'Cambridge'):
      temp = weights['cambridge']
      weights['cambridge'] = int(temp) + 1
    elif(last_visited == 'Merriam Webster'):
        temp = weights['merriam']
        weights['merriam'] = int(temp) + 1
    elif(last_visited == 'Famous Birthdays'):
        temp = weights['famous']
        weights['famous'] = int(temp) + 1
    elif(last_visited == 'Wikipedia'):
        temp = weights['wikipedia']
        weights['wikipedia'] = int(temp) + 1
    elif(last_visited == 'Urban'):
        temp = weights['urban']
        weights['urban'] = int(temp) + 1
    # Update weight
    write_text = open("weights.txt","w+")
    for key, value in weights.items():
      write_text.write(str(key) + ' ' + str(value) + '\n')
    write_text.close()
  return new_clue
```

```
 except:
   print('There is a problem related to ' + solution.upper())

# API Calls
@puzzle.route('/get_old_clues_across')
def get_old_clues_across():
 redirect_puzzle_page()
 across, _ = inspect_old_clues()
 return jsonify(across)

@puzzle.route('/get_old_clues_down')
def get_old_clues_down():
 redirect_puzzle_page()
 _, down = inspect_old_clues()
 return jsonify(down)

@puzzle.route('/get_puzzle_layout')
def get_puzzle_layout():
 cell_types = dict(enumerate(['regular'] * 25))
 redirect_puzzle_page()
 for i in inspect_puzzle_layout():
   cell_types[i] = 'block'
 return jsonify(cell_types)

@puzzle.route('/get_cell_numbers')
def get_cell_numbers():
 redirect_puzzle_page()
 cell_numbers = inspect_cell_numbers()
 return jsonify(cell_numbers)

@puzzle.route('/get_solutions')
def get_solutions():
 redirect_puzzle_page()
 reveal_solutions()
 solution_dict = {}
 solution_list = inspect_solutions()
 for i in range(0, 25):
   solution_dict[str(i)] = solution_list[i]
 return jsonify(solution_dict)

@puzzle.route('/generate_new_clues')
def generate_new_clues():
 redirect_puzzle_page()
 reveal_solutions()
 solution_dict = convert_solutions(inspect_solutions())
 new_clues = {}
 for i in solution_dict:
   new_clues[i] = find_alt_clue(solution_dict[i])
 return jsonify(new_clues)

# Unused API
@puzzle.route('/get_solutions_dict')
def get_solutions_dict():
 redirect_puzzle_page()
 reveal_solutions()
 solution_dict = convert_solutions(inspect_solutions())
 return jsonify(solution_dict)

open_browser()
```

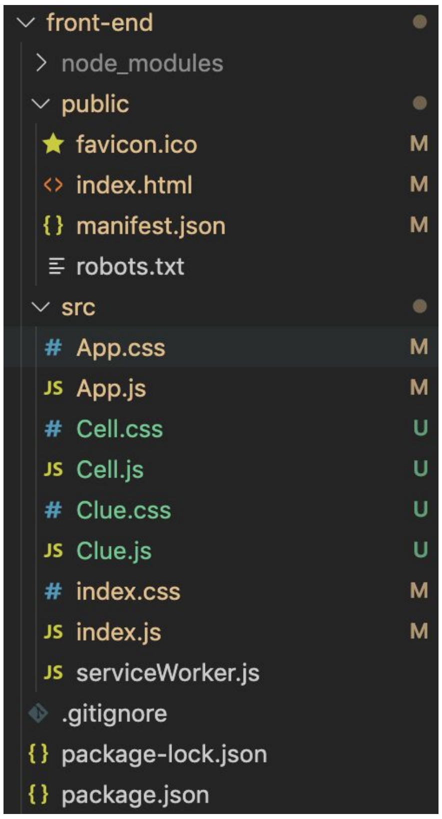
Pipfile

```
[[source]]
name = "pypi"
url = "https://pypi.org/simple"
verify_ssl = true

[dev-packages]

[packages]
flask = "*"
beautifulsoup4 = "*"
flask-cors = "*"
selenium = "*"

[requires]
python_version = "3.7"
```

weights.txt

```
wordnet 63
merriam 21
cambridge 20
famous 12
wiki 3
urban 1
```

A2: Front-end



```
∨ front-end                 ●
  > node_modules
  ∨ public                  ●
     favicon.ico            M
     index.html             M
     manifest.json          M
     robots.txt
  ∨ src                     ●
     App.css                M
     App.js                 M
     Cell.css               U
     Cell.js                U
     Clue.css               U
     Clue.js                U
     index.css              M
     index.js               M
     serviceWorker.js
  .gitignore
  package-lock.json
  package.json
```

Create a React App (see this link https://github.com/facebook/create-react-app) and copy package.json file. Update node modules using npm.

Copy JS and CSS files.

Type $ npm start

package.json

```
{
 "name": "front-end",
 "version": "0.1.0",
 "private": true,
 "dependencies": {
   "react": "^16.12.0",
   "react-dom": "^16.12.0",
   "react-scripts": "3.2.0"
 },
 "scripts": {
   "start": "react-scripts start",
   "build": "react-scripts build",
   "test": "react-scripts test",
   "eject": "react-scripts eject"
 },
 "eslintConfig": {
   "extends": "react-app"
 },
 "browserslist": {
   "production": [
     ">0.2%",
     "not dead",
     "not op_mini all"
   ],
   "development": [
     "last 1 chrome version",
     "last 1 firefox version",
     "last 1 safari version"
   ]
 }
}
```

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="logo192.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <title>CS 461 - ARTIFICIAL INTELLIGENCE PROJECT</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';
```

```
ReactDOM.render(<App />, document.getElementById('root'));

// If you want your app to work offline and load faster, you can change
// unregister() to register() below. Note this comes with some pitfalls.
// Learn more about service workers: https://bit.ly/CRA-PWA
serviceWorker.unregister();
```

index.css

```
body {
   font-family: 'Times';
}

a, abbr, acronym, address, applet, article, aside, audio, b, big, blockquote, body,
canvas, caption, center, cite, code, dd, del, details, dfn, div, dl, dt, em, embed,
fieldset, figcaption, figure, footer, form, h1, h2, h3, h4, h5, h6, header, hgroup,
html, i, iframe, img, ins, kbd, label, legend, li, mark, menu, nav, object, ol, output,
p, pre, q, ruby, s, samp, section, small, span, strike, strong, sub, summary, sup,
table, tbody, td, tfoot, th, thead, time, tr, tt, u, ul, var, video {
 margin: 0;
 padding: 0;
 border: 0;
 font-size: 100%;
 font: inherit;
 vertical-align: baseline;
}
```

App.js

```
import React, { Component } from "react";
import "./App.css";
import Clue from "./Clue";
import Cell from "./Cell";

class App extends Component {
 constructor(props) {
   super(props)

   this.state = {
       date : new Date(),
       old_across : [],
       old_down : [],
       puzzle_layout : [],
       cell_numbers : [],
       solutions : [],
       new_across : [],
       new_down : []
   }
 }
 componentDidMount() {
   fetch("http://localhost:5000/get_old_clues_across").then((res) => {
     res.json().then((clues) => {
       let temp = []
       Object.keys(clues).forEach((key) => {
         temp = [...temp, { 'key' : key + 'A', 'clue' : clues[key] }]
       })
       this.setState({
         old_across : temp
       }, () => {
         fetch("http://localhost:5000/get_old_clues_down").then((res) => {
           res.json().then((clues) => {
             let temp = []
             Object.keys(clues).forEach((key) => {
               temp = [...temp, { 'key' : key + 'D', 'clue' : clues[key] }]
             })
             this.setState({
               old_down : temp
             }, () => {
               fetch("http://localhost:5000/get_puzzle_layout").then((res) => {
                 res.json().then((types) => {
                   let temp = []
                   Object.keys(types).forEach((key) => {
                     temp = [...temp, types[key]]
                   })
                   this.setState({
                     puzzle_layout : temp
                   }, () => {
                     fetch("http://localhost:5000/get_cell_numbers").then((res) => {
                       res.json().then((numbers) => {
                         this.setState({
                           cell_numbers : numbers
                         }, () => {
                           fetch("http://localhost:5000/get_solutions").then((res) => {
                             res.json().then((solutions) => {
                               let temp = []
                               Object.keys(solutions).forEach((key) => {
                                 temp = [...temp, solutions[key]]
                               })
                               this.setState({
                                 solutions : temp
                               }, () => {

fetch('http://localhost:5000/generate_new_clues').then((res) => {
                                 res.json().then((clues) => {
                                   let temp_across = []
                                   let temp_down = []
                                   Object.keys(clues).forEach((key) => {
                                     if(key.substring(1) === 'A') {
                                       temp_across = [...temp_across, { 'key' : key,
'clue' : clues[key] }]

                                     }
                                     else if(key.substring(1) === 'D') {
                                       temp_down = [...temp_down, { 'key' : key,
'clue' : clues[key] }]
                                     }
                                   })
                                   this.setState({
                                     new_across : temp_across,
                                     new_down : temp_down
                                   })
                                 })
                               })
                             })
                           })
                         })
                       })
                     })
                   })
                 })
               })
             })
           })
         })
       })
     })
   })
 }

 render() {
   let cell_number = 1
   return (
     <div className="App">
       <article className="Puzzle-Layout" aria-label="Main Puzzle Layout">
         <section className="ClueBar-And-Board" aria-label="Game Board with Clue Bar">
           <section className="Board" aria-label="Game Board">
             <div className="Board-Content" style={{ transition: "transform 0s ease 0s",
transform: "translate(0px, 0px) scale(1)", touchAction: "none", userSelect: "none",
WebkitUserDrag: "none", WebkitTapHighlightColor: "rgba(0, 0, 0, 0)" }}>
               <svg id="board" className="Board-SVG" preserveAspectRatio="xMidYMin meet"
aria-labelledby="boardTitle" aria-describedby="boardDesc"
xmlns="http://www.w3.org/2000/svg" viewBox="0 0 506.00 564.00">
                 <title id="boardTitle">Puzzle Board</title>
                 <desc id="boardDesc">Game Board for the Crossword</desc>
                 <g data-group="cells" role="table">
                   {
                     this.state.puzzle_layout.map((item, i) => {
                       return <Cell key={i} index={i} type={item}
textAnchorStart={this.state.cell_numbers.includes(i) ? cell_number++ : ''}
textAnchorMiddle={this.state.solutions[i]}/>
                     })
                   }
                 </g>
                 <g data-group="grid">
                   <path d="M3.00,103.00 l500.00,0.00 M3.00,203.00 l500.00,0.00
M3.00,303.00 l500.00,0.00 M3.00,403.00 l500.00,0.00 M103.00,3.00 l0.00,500.00
M203.00,3.00 l0.00,500.00 M303.00,3.00 l0.00,500.00 M403.00,3.00 l0.00,500.00"
```

```
stroke="dimgray" vectorEffect="non-scaling-stroke"></path>
                   <rect x="1.50" y="1.50" width="503.00" height="503.00" stroke="black"
strokeWidth="3.00" fill="none"></rect>
                 </g>
                 <text x="506.00" y="564.50" textAnchor="end" fontSize="30" style={{
fontFamily : "Times" }}>
                   {(this.state.date.getDate() < 10 ? ('0' + this.state.date.getDate())
: this.state.date.getDate()) +
                   '.' + String((parseInt(this.state.date.getMonth()) + 1) < 10 ?
                   ('0' + parseInt(this.state.date.getMonth() + 1)) :
parseInt(this.state.date.getMonth() + 1)) +
                   '.' + this.state.date.getFullYear()}
                 </text>
                 <text x="506.00" y="534.50" textAnchor="end" fontSize="30" style={{
fontFamily : "Times" }}>
                   FIRING JOEL FAGLIANO
                 </text>
               </svg>
             </div>
           </section>
         </section>
         <section className="ClueList-Layout">
           <div className="ClueList-Wrapper">
             <h3 className="ClueList-Title">Across</h3>
             <ol className="ClueList-List">
               {
                 this.state.old_across.map((item) => {
                   return <Clue key={item.key} number={item.key.charAt(0)}
clue={item.clue}/>
                 })
               }
             </ol>
           </div>
           <div className="ClueList-Wrapper">
             <h3 className="ClueList-Title">New Across</h3>
             <ol className="ClueList-List">
               {
                 this.state.new_across.map((item) => {
                   return <Clue key={item.key} number={item.key.charAt(0)}
clue={item.clue}/>
                 })
               }
             </ol>
           </div>
           <div className="ClueList-Wrapper">
             <h3 className="ClueList-Title">Down</h3>
             <ol className="ClueList-List">
               {
                 this.state.old_down.map((item) => {
                   return <Clue key={item.key} number={item.key.charAt(0)}
clue={item.clue}/>
                 })
               }
             </ol>
```

```
          </div>
          <div className="ClueList-Wrapper">
            <h3 className="ClueList-Title">New Down</h3>
            <ol className="ClueList-List">
              {
                this.state.new_down.map((item) => {
                  return <Clue key={item.key} number={item.key.charAt(0)}
clue={item.clue}/>
                })
              }
            </ol>
          </div>
        </section>
      </article>
    </div>
  );
 }
}

export default App;
```

App.css

```
.Puzzle-Layout {
 -webkit-box-align: start;
 -webkit-align-items: flex-start;
 -ms-flex-align: start;
 align-items: flex-start;
 height: 100vh;
 max-height: 660px;
 -webkit-justify-content: space-around;
 -ms-flex-pack: distribute;
 justify-content: space-around;
 margin: auto;
 max-width: 1132px;
 padding: 10px;
 margin-top: 50px;
}

.Puzzle-Layout>* {
 height: 100%;
 width: 48%;
}

.Puzzle-Layout, .Puzzle-Layout>* {
 display: -webkit-box;
 display: -webkit-flex;
 display: -ms-flexbox;
 display: flex;
}

.ClueBar-And-Board {
 -webkit-box-orient: vertical;
 -webkit-box-direction: normal;
 -webkit-flex-direction: column;
 flex-direction: column;
}

.ClueBar {
 -webkit-box-align: center;
 -webkit-align-items: center;
 -ms-flex-align: center;
 align-items: center;
 background: #dcefff;
 display: -webkit-box;
 display: -webkit-flex;
 display: -ms-flexbox;
 display: flex;
 font-size: 1.2em;
 margin-bottom: 10px;
 min-height: 60px;
}

.ClueBar-Number {
 font-weight: 700;
 min-width: 60px;
 text-align: center;
 width: 60px;
}

.ClueBar-Text {
 padding-right: 10px;
 line-height: 23px;
}

.Board {
 width: 100%;
 -webkit-box-flex: 1;
 -webkit-flex-grow: 1;
 flex-grow: 1;
 position: relative;
}

.Board-Content {
 position: absolute;
 top: 1px;
 left: 1px;
 right: 1px;
 bottom: 1px;
}

svg:not(:root) {
 overflow: hidden;
}

.Board-SVG {
 display: block;
 font-family: arial,sans-serif;
 -webkit-user-select: none;
 -moz-user-select: none;
 user-select: none;
 width: 100%;
 height: 100%;
}

.ClueList-Layout {
 -webkit-flex-wrap: wrap;
 -ms-flex-wrap: wrap;
 flex-wrap: wrap;
 -webkit-justify-content: space-around;
 -ms-flex-pack: distribute;
 justify-content: space-around;
 max-height: 100%;
}

.ClueList-Wrapper {
 width: 47%;
 min-height: 50%;
}

.ClueList-Title {
 border-bottom: 1px solid #e6e6e6;
 content: attr(name);
 font-weight: 700;
 line-height: 2;
 text-transform: uppercase;
}

.ClueList-List {
 height: calc(100% - 30px);
 overflow-y: scroll;
}
```

Cell.js

```
import React, { Component } from 'react';
import "./Cell.css";

class Cell extends Component {
 render() {
   return (
     <g>
       {
         this.props.type === 'block' &&
         <rect role="cell" className={"Cell-Block"} x={(this.props.index % 5) * 100 + 3}
y={~~(this.props.index / 5) * 100 + 3} width="100.00" height="100.00"></rect>
       }
       {
         this.props.type === 'regular' &&
         <rect role="cell"  className={"Cell-Cell"} x={(this.props.index % 5) * 100 + 3}
y={~~(this.props.index / 5) * 100 + 3} width="100.00" height="100.00"></rect>
       }
       {
         this.props.textAnchorStart &&
         <text x={(this.props.index % 5) * 100 + 5} y={~~(this.props.index / 5) * 100 +
36.83} textAnchor="start" fontSize="33.33">{this.props.textAnchorStart}</text>
       }
       {
         this.props.textAnchorMiddle &&
         <text x={(this.props.index % 5) * 100 + 53} y={~~(this.props.index / 5) * 100 +
94.67} textAnchor="middle" fontSize="66.67">{this.props.textAnchorMiddle}</text>
       }
     </g>
   )
 }
}

export default Cell;
```

Cell.css

```
.Cell-Cell {
 fill: #fff;
}

.Cell-Block {
 fill: #000;
}

text {
 -webkit-transition: font-size .1s;
 transition: font-size .1s;
}
```

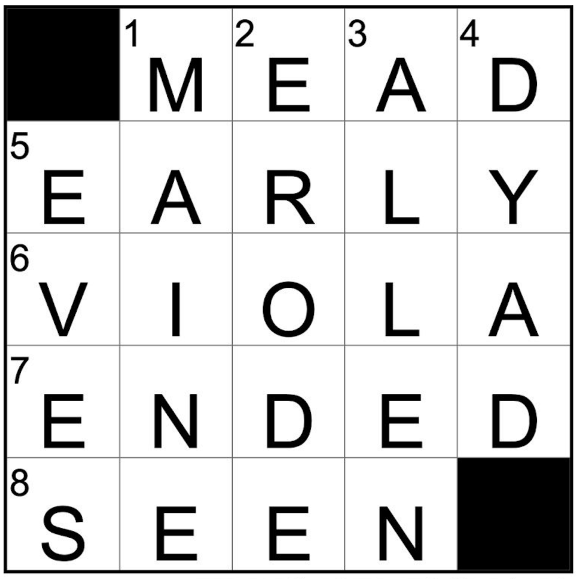
Clue.js

```
import React, { Component } from 'react';
import "./Clue.css";

class Clue extends Component {
 render() {
   return (
     <li className="Clue-Li">
       <span className="Clue-Label">
         {this.props.number}
       </span>
       <span className="Clue-Text">
         {this.props.clue}
       </span>
     </li>
   )
 }
}

export default Clue;
```

Clue.css

```
.Clue-Li {
 border-left: 10px solid transparent;
 cursor: pointer;
 display: -webkit-box;
 display: -webkit-flex;
 display: -ms-flexbox;
 display: flex;
 padding: 5px 1px;
}

.Clue-Li>span {
 line-height: 1.4;
 margin-right: 5px;
}

.Clue-Label {
 font-weight: 700;
 margin-right: 10px;
 text-align: right;
 min-width: 24px;
 width: 24px;
}

.Clue-Text {
 margin-left: 5px;
}
```

Appendix B: Screenshots



■	1 M	2 E	3 A	4 D
5 E	A	R	L	Y
6 V	I	O	L	A
7 E	N	D	E	D
8 S	E	E	N	■

FIRING JOEL FAGLIANO
03.12.2019

ACROSS

1 Fermented honey drink

5 How guests at a surprise party should arrive

6 ___ Davis, first black actress to win an Oscar, Emmy and Tony

7 Finished up

8 Spotted

DOWN

1 Home to the colleges Bates and Colby

2 Slowly wear away

3 Poet Ginsberg who wrote "Howl"

4 Group of two

5 Two big nights in December

NEW ACROSS

1 A fermented beverage made of water and honey, malt, and yeast

5 Near the beginning of a period of time

6 Acclaimed actress of stage, TV, and film who won Tony Awards for her performances in King Hedly II and Fences

7 Have an end, in a temporal, spatial, or quantitative sense

8 Perceive by sight or have the power to perceive by sight

NEW DOWN

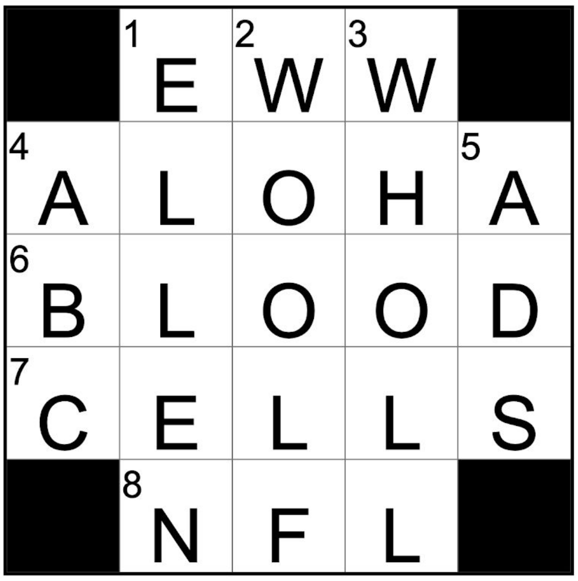
1 A state in New England

2 Become ground down or deteriorate

3 Star shooting guard for the Philadelphia 76ers from 1996 to 2006

4 Two items of the same kind

5 God created Eve from Adam's rib and placed Adam and Eve in the Garden of Eden



■	1 E	2 W	3 W	■
4 A	L	O	H	5 A
6 B	L	O	O	D
7 C	E	L	L	S
■	8 N	F	L	■

FIRING JOEL FAGLIANO
04.12.2019

ACROSS

1 "Gross!"

4 Honolulu hello

6 With 7-Across, your body has produced millions of them while you've been reading this clue

7 See 6-Across

8 Org. with three Thanksgiving Day games

DOWN

1 Talk show host who voices Pixar's Dory

2 Virginia who wrote "Mrs. Dalloway"

3 "___ Pull Santa's Sleigh Tonight?" (children's book)

4 Epitome of simplicity

5 What're found on the sides of many buses

NEW ACROSS

1 An expression of disgust

4 Used as a greeting or farewell

6 The red liquid that is sent around the body by the heart, and carries oxygen and important substances to organs and tissue, and removes waste products:

7 Any small compartment

8 The main organized group of US football teams that have players who are paid for playing

NEW DOWN

1 Comedian who starred on the sitcom Ellen in the '90s then went on to host the popular talk show The Ellen DeGeneres Show

2 Prominent member of the Bloomsbury Group (1882 1941)

3 Who'll be at the party tomorrow?

4 The elementary stages of any subject (usually plural)

5 A public promotion of some product or service

1 S	2 T	3 A	4 F	5 F
6 M	A	R	L	A
7 A	B	B	O	T
8 C	L	O	U	T
9 K	E	R	R	Y

FIRING JOEL FAGLIANO
10.12.2019

ACROSS

1 Lines on a music score

6 ___ Maples, second wife of Donald Trump

7 Monk's superior

8 Political influence

9 Democratic nominee of 2004

DOWN

1 Hit with an open palm

2 Wedding reception grouping

3 ___ Day, celebration on the last Friday in April

4 Common baking ingredient

5 Like many foods avoided on diets

NEW ACROSS

1 Personnel who assist their superior in carrying out an assigned task

6 Both an Instagram personality and YouTube content creator, she has accumulated more than 800,000 followers on the former platform

7 The superior of an abbey of monks

8 A target used in archery

9 Actress who starred on the popular ABC show Scandal and played Nikki in I Think I Love My Wife with Chris Rock

NEW DOWN

1 A blow from a flat object (as an open hand)

2 A set of data arranged in rows and columns

3 Tree (as opposed to shrub)

4 Fine powdery foodstuff obtained by grinding and sifting the meal of a cereal grain

5 A rotund individual

■	1 S	2 U	3 I	4 T
5 D	U	N	N	O
6 A	S	P	E	N
7 S	H	I	R	E
8 H	I	N	T	■

FIRING JOEL FAGLIANO
17.12.2019

ACROSS

1 Spades, hearts, diamonds or clubs

5 "Beats me"

6 Colorado ski resort

7 Frodo and Bilbo's home, with "the"

8 "It rhymes with mint," for this answer

DOWN

1 Seaweed-wrapped roll

2 Remove, as a corsage

3 Chemically nonreactive

4 Voicemail prompt

5 "Now ___ away! ___ away! ___ away all!" (line in "A Visit From St. Nicholas")

NEW ACROSS

1 A set of garments such as

5 Used in writing to represent the sound of the phrase (I) don't know when it is spoken quickly

6 Instagram model and influencer who is best recognized for her affiliation with the boy band Why Don't We

7 A former administrative district of England

8 An indirect suggestion

NEW DOWN

1 Rice (with raw fish) wrapped in seaweed

2 Remove the pins from

3 Unable to move or resist motion

4 R&B singer who is best known as comprising one-third of the pop-R&B trio SJ3

5 Well known for his series regular role on the 2014 show Summer Break, he has guest-starred on notable series such as ABC's Modern Family, Nickelodeon's AwesomenessTV, and Adult Swim's Children's Hospital

■	1 S	2 H	3 O	4 E
■	5 H	O	W	L
6 T	R	U	N	K
7 F	U	S	E	■
8 A	G	E	D	■

FIRING JOEL FAGLIANO
20.12.2019

ACROSS

1 Something bought and soled
5 Sound of the roaring wind
6 Part of a car, tree or elephant
7 Lit part of a stick of dynamite
8 Like fine wines

DOWN

1 Apathetic gesture
2 Group that voted for Trump's impeachment
3 Possessed
4 Colorado has the largest population of this animal in the world, at over 280,000
6 Nonprofit that recruits college grads into education: Abbr.

NEW ACROSS

1 One of a pair of coverings for your feet, usually made of a strong material such as leather, with a thick leather or plastic sole
5 A long loud emotional utterance
6 The main stem of a tree
7 To reduce to a liquid or plastic state by heat
8 People who are old collectively

NEW DOWN

1 A gesture involving the shoulders
2 A dwelling that serves as living quarters for one or more families
3 Have ownership or possession of
4 Large northern deer with enormous flattened antlers in the male
6 A branch of the Queensland Police Service, responsible for the investigation of online child exploitation and abuse

■	■	1 F	2 O	3 X
4 S	5 P	E	R	M
6 K	O	R	E	A
7 E	U	R	O	S
8 W	R	Y	■	■

FIRING JOEL FAGLIANO
24.11.2019

ACROSS

1 Animal whose babies are known as "kits"
4 What fuses with an ovum
6 Peninsula divided at the 38th parallel
7 Currency symbols that can be typed using Ctrl+Alt+E
8 Like a knowing smirk

DOWN

1 One way to get to Staten Island
2 Best-selling cookie brand in the U.S.
3 Present day, for short
4 Distort
5 Rain heavily

NEW ACROSS

1 Alert carnivorous mammal with pointed muzzle and ears and a bushy tail
4 The male reproductive cell
6 An Asian peninsula (off Manchuria) separating the Yellow Sea and the Sea of Japan
7 The basic monetary unit of most members of the European Union (introduced in 1999)
8 Humorously sarcastic or mocking

NEW DOWN

1 Dutch music producer and DJ, known for his weekly radio show, Corsten's Countdown
2 Chocolate cookie with white cream filling
3 A Christian holiday celebrating the birth of Christ
4 Turn or place at an angle
5 Cause to run

■	1 P	2 O	3 S	4 T
5 P	L	U	T	O
6 H	A	S	A	N
7 E	N	T	R	Y
8 W	E	S	T	■

FIRING JOEL FAGLIANO
25.11.2019

ACROSS

1 Share on social media

5 Mickey Mouse's dog

6 Comedian Minhaj who hosts "Patriot Act"

7 Contest submission

8 Geopolitical term for Europe and the Americas, with "the"

NEW ACROSS

1 The position where someone (as a guard or sentry) stands or is assigned to stand

5 A cartoon character created by Walt Disney

6 Actor and comedian most well known for serving as a senior correspondent on Comedy Central's The Daily Show

7 An item inserted in a written record

8 The countries of (originally) Europe and (now including) North America and South America

DOWN

1 What Superman is mistaken for

2 Removes from power

3 Triangular button on a video game controller

4 Broadway award

5 "That was a close one!"

NEW DOWN

1 An aircraft that has a fixed wing and is powered by propellers or jets

2 Remove from a position or office

3 The beginning of anything

4 American dancer who is one half of the popular dance duo The Lopez Brothers

5 Used to express relief.

1 P	2 S	3 S	4 T	■
5 T	I	T	H	6 E
7 S	N	A	R	L
8 D	A	N	E	S
■	9 I	D	E	A

FIRING JOEL FAGLIANO
26.11.2019

ACROSS

1 "Hey, you! Over here!"

5 Give 10% to the church

7 Hair tangle

8 "Great" dogs

9 Inkling

NEW ACROSS

1 "____! i'm over here."

5 A levy of one tenth of something

7 A vicious angry growl

8 A native or inhabitant of Denmark

9 The content of cognition

DOWN

1 Condition for a returning combat vet

2 Mount where Moses received the Ten Commandments

3 Roadside produce seller

4 Number of bones in the human ear

6 Queen in "Frozen 2"

NEW DOWN

1 A mental condition in which a person suffers severe anxiety and depression after a very frightening or shocking experience, such as an accident or a war

2 It is believed to be the peak on which Moses received the Ten Commandments

3 A support or foundation

4 The cardinal number that is the sum of one and one and one

6 Swedish fashion model, most recognized for her work with designers like Lilly Pulitzer and Dior

1 F	2 A	3 C	4 T	■
5 A	L	I	B	6 I
7 V	E	G	A	N
8 E	X	A	L	T
■	9 A	R	L	O

FIRING JOEL FAGLIANO
26.12.2019

ACROSS

1 Bit of trivia

5 Courtroom excuse

7 Like the Impossible Burger

8 Hold in high regard

9 Folk singer Guthrie

DOWN

1 Top choice, slangily

2 Inconvenient name for an Amazon Echo owner

3 Trademark item for Groucho Marx or Winston Churchill

4 Lowest level of Little League

6 Enamored with

NEW ACROSS

1 A piece of information about circumstances that exist or events that have occurred

5 The plea of having been at the time of the commission of an act elsewhere than at the place of commission

7 A strict vegetarian

8 Praise, glorify, or honor

9 Son of Woody Guthrie who sang "Alice's Restaurant Massacree" in 1966

NEW DOWN

1 Informal for favourite

2 YouTube vlogger who is known for her self-titled channel

3 A roll of tobacco for smoking

4 Baseball modified for youngsters in which the ball is batted from a tee of adjustable height rather than being pitched

6 To the inside or middle of a place, container, area, etc.

■	■	1 C	2 E	3 O
4 T	5 H	I	G	H
6 W	I	N	G	S
7 A	N	D	S	O
8 S	T	Y	■	■

FIRING JOEL FAGLIANO
28.11.2019

ACROSS

1 Top of a corporate ladder

4 Certain piece of turkey

6 Some pieces of turkey

7 "Therefore ..."

8 Totally messy room

DOWN

1 ___ Lou Who, character in "How the Grinch Stole Christmas"

2 Food sold in sixes and dozens

3 Very

4 "___ the night before Christmas ..."

5 Crossword puzzle clue, for example

NEW ACROSS

1 The corporate executive responsible for the operations of the firm

4 The part of the leg between the hip and the knee

6 A movable organ for flying (one of a pair)

7 An unnamed or unspecified person, thing, or action

8 An infection of the sebaceous gland of the eyelid

NEW DOWN

1 Instagram sensation who gained fame after Justin Bieber posted a photo of her on his feed and asked "Omg who is this!!!"

2 Especially the thin shelled reproductive body laid by e.g. female birds

3 Used in sarcastic manner to comment on a personality trait or to insult someone who is bragging about

4 The Old English combination of the words "it" and "was", now creeping back into modern culture

5 An indirect suggestion