Q1-4. [16 points, *answer should be in your PDF submission*] An alternate measure of economic rank of an area might be the inverse of a "hardship index," defined and quantified here. How well, or poorly, do your calculations of TrafficRank for Chicago community areas correspond with the inverse of hardship index? Give *a qualitative analysis in plain English*.

Comparing TrafficRank results with the hardship index reveals both alignments and significant discrepancies:

1. Notable Alignments:
    - Region 7 (Lincoln Park) ranks highest in TrafficRank and shows low hardship index, which is consistent with its known economic prosperity
    - The correlation is limited to select areas, suggesting TrafficRank captures only certain aspects of economic well-being
2. Key Discrepancies:
    - Region 8 (Near North Side) shows surprisingly low TrafficRank despite being one of Chicago's most affluent areas with low hardship
    - Most of the top TrafficRank areas (31, 27, 5, 75) don't strongly correlate with areas of low hardship
    - Similarly, the bottom TrafficRank regions (73, 71, 51, 17) don't consistently match high-hardship areas
3. Explanatory Factors:
    - Transportation alternatives: Affluent areas may rely more on private vehicles or other transit options
    - Land use patterns: High TrafficRank might indicate commercial/industrial zones rather than residential wealth
    - Infrastructure: Areas with major transportation hubs or business districts may show high taxi traffic regardless of local economic conditions
    - Commuting patterns: High taxi usage might reflect worker movement rather than resident economic status

This analysis suggests TrafficRank is more indicative of an area's role in the city's transportation and commercial network than its economic prosperity as measured by the hardship index.

Q2-1. [4 points*, answer should be in your PDF submission*] We can calculate TF values of each word in a given document. Explain why the calculation of IDF can only apply to a corpus, not to a specific document.

TF (Term Frequency) can be calculated for a single document because it measures how frequently a term appears within that specific document. However, IDF (Inverse Document Frequency) can only be calculated across a corpus because it specifically measures how unique or rare a term is across multiple documents. The IDF calculation requires knowing the total number of documents in the corpus and the number of documents containing each term - log(N/df), where N is the total number of documents and df is the number of documents containing the term. This ratio is impossible to calculate with just one document, as it inherently requires comparing term occurrence patterns across multiple documents to determine a term's discriminative power within the entire collection.

Q2-2. [6 points, *answer should be in your PDF submission*] The implementation of Scikit-Learn's IDF calculation differs from that of the "standard" calculation. What is the justification for this change?

The difference between Scikit-Learn's IDF calculation and the standard formula lies in where the '1' is added:

Scikit-Learn: idf(t) = log[n/df(t)] + 1 Standard: idf(t) = log[n/(df(t) + 1)]

Scikit-Learn's modification has an important practical justification: it prevents terms that appear in all documents (where df(t) = n) from being completely ignored in the analysis. In the standard formula, when a term appears in all documents, the calculation becomes log(1) = 0, effectively eliminating that term's influence. However, in Scikit-Learn's version, even when df(t) = n, the formula gives 1 (since log(1) + 1 = 1), ensuring that commonly occurring terms still retain some weight in the final TF-IDF score.

This modification is particularly useful in real-world applications where common terms might still carry important semantic meaning despite appearing in all documents. It provides a more nuanced approach to term weighting rather than completely discarding ubiquitous terms.

Q2-3. [25 points, *answer should be in your PDF submission*] Each president has a .tar.gz file containing his speeches. Filter the text using remove_stopwords(bow) code provided[1], where bow is the bag of words in the text. Write a Python procedure to calculate TF.IDF for any president's speeches and output the top-15 most important words in their speech.

The top-15 most important word for president Hoover is as following:

```
Top 15 words for hoover:
government: 0.0120
000: 0.0103
people: 0.0080
economic: 0.0076
federal: 0.0066
american: 0.0064
states: 0.0059
business: 0.0055
country: 0.0052
public: 0.0052
tariff: 0.0050
congress: 0.0049
democratic: 0.0048
great: 0.0047
national: 0.0044
```

1

```
import requests
stop_url = "https://gist.githubusercontent.com/rg089/35e00abf8941d72d419224cfd5b5925d/raw/12d899b70156fd0041fa9778d657330b024b959c/stopwords.txt"
stopwords_list = requests.get(stop_url).content
stopwords = set(stopwords_list.decode().splitlines())
stopwords = list(stopwords)

def remove_stopwords(low): # low = list of words
    list_ = re.sub(r"[^a-zA-Z0-9]", " ", low.lower()).split()
    return [itm for itm in list_ if itm not in stopwords]
```

```
import numpy as np
from collections import Counter
import glob
import re
import requests
import os

def calculate_president_tfidf(target_president):
    # Dictionary to store all speeches for each president
    president_speeches = {}

    # Read all speeches for each president
    for president_dir in os.listdir('.'):
        if os.path.isdir(president_dir):
            all_speeches = []
            for filepath in glob.glob(f"{president_dir}/*.txt"):
                with open(filepath, 'r', encoding='utf-8') as f:
                    speech = f.read()
                    filtered_speech = remove_stopwords(speech)
                    all_speeches.extend(filtered_speech)
            president_speeches[president_dir] = all_speeches

    # Check if target president exists
    if target_president not in president_speeches:
        raise ValueError(f"President {target_president} not found in directory")

    # Calculate document frequencies
    doc_freq = Counter()
    for president, speeches in president_speeches.items():
        unique_words = set(speeches)
        for word in unique_words:
            doc_freq[word] += 1

    # Calculate IDFs
    N = len(president_speeches)
    idfs = {}
    for word, df in doc_freq.items():
        idfs[word] = np.log((N + 1)/(df + 1)) + 1

    # Calculate TF-IDF for target president
    speeches = president_speeches[target_president]
    total_words = len(speeches)
```

```
  word_counts = Counter(speeches)

  # Calculate TF-IDF scores
  tfidf_scores = {}
  for word, count in word_counts.items():
      tf = count/total_words
      tfidf_scores[word] = tf * idfs[word]

  # Get top 15 words
  top_words = sorted(tfidf_scores.items(),
                     key=lambda x: x[1],
                     reverse=True)[:15]

  return top_words
```

Q2-4. [15 points, *answer should be in your PDF submission*] Examine the result carefully – at least ***some*** of the top TF.IDF words should be historically consistent with what was going on in the country at the time[2]. You just have *very slight* control over the outcome (by adding more words to the initial set of stopwords).

Analysis of the TF-IDF results shows historically significant terms that accurately reflect the challenges of Hoover's presidency (1929-1933). The high frequency of terms like 'government', 'economic', 'business', and 'tariff' aligns with the onset of the Great Depression and Hoover's response to it. Particularly, the prominence of 'tariff' likely relates to the controversial Smoot-Hawley Tariff Act of 1930, which raised import duties to protect American businesses but ultimately worsened the economic crisis. The term 'government' is significant as it reflects Hoover's evolving approach - while initially favoring voluntary cooperation between business and government, he eventually moved toward more direct government intervention through programs like the Reconstruction Finance Corporation. These terms capture the tension between Hoover's initial laissez-faire philosophy and the mounting pressure for federal intervention during the Depression.

To improve the TF-IDF analysis, additional stopwords could include common administrative terms like 'department', 'secretary', and 'federal' to better highlight historically distinctive terms specific to Hoover's economic policies and Depression-era responses.

[2] The main point of this sub-question is to emphasize that our code will always give us *some* answer; *our challenge is to verify that the answer "makes sense."*