Natural Language Processing-5

Narasimha Rohit Katta

April 2024

1

A.TF-IDF Calculation for the word ”coffee”

(a) Calculate IDF for ”coffee”:

$$IDF_{\text{coffee}} = \log_{10}\left(\frac{N}{df_{\text{coffee}}}\right) = \log_{10}\left(\frac{4}{2}\right) = \log_{10}(2) \approx 0.3010$$

(b) Calculate TF for ”coffee” in Document 1:

$$TF_{\text{coffee, doc1}} = \log_{10}\left(\text{count}(\text{"coffee"}, \text{doc1}) + 1\right) = \log_{10}(26) \approx 1.4149$$

(c) Calculate TF-IDF for ”coffee” in Document 1:

$$TFIDF_{\text{coffee, doc1}} = TF_{\text{coffee, doc1}} \times IDF_{\text{coffee}} = 1.4149 \times 0.3010 \approx 0.4258$$

Similarly, for Document 2:
(b) Calculate TF for ”coffee” in Document 2:

$$TF_{\text{coffee, doc2}} = \log_{10}\left(\text{count}(\text{"coffee"}, \text{doc2}) + 1\right) = \log_{10}(19) \approx 1.2787$$

(c) Calculate TF-IDF for ”coffee” in Document 2:

$$TFIDF_{\text{coffee, doc2}} = TF_{\text{coffee, doc2}} \times IDF_{\text{coffee}} = 1.2787 \times 0.3010 \approx 0.3848$$

For Document 3, since ”coffee” does not appear, the TF is 0, leading to a TF-IDF value of 0:

$$TF_{\text{coffee, doc3}} = \log_{10}(1) = 0$$

$$TFIDF_{\text{coffee, doc3}} = 0 \times 0.3010 = 0$$

Similarly, for Document 4, with no occurrences of ”coffee,” the TF and TF-IDF values are 0:

$$TF_{\text{coffee, doc4}} = \log_{10}(1) = 0$$

$$TFIDF_{\text{coffee, doc4}} = 0 \times 0.3010 = 0$$

(b) TF-IDF Calculation for the term ”vacation”:

For the term ”vacation”, we’ll calculate the Term Frequency-Inverse Document Frequency (TF-IDF) across four documents. Given that the term appears in all four documents ($N = 4$ and $df_{\text{vacation}} = 4$), the Inverse Document Frequency (IDF) is calculated as follows:

$$IDF_{\text{vacation}} = \log_{10}\left(\frac{N}{df_{\text{vacation}}}\right) = \log_{10}\left(\frac{4}{4}\right) = \log_{10}(1) = 0$$

Since the IDF is 0, all TF-IDF values will be 0 regardless of the TF values.

Document 1: For Document 1, the term frequency of ”vacation” is:

$$TF_{\text{vacation, doc1}} = \log_{10}(count(\text{vacation, doc1})+1) = \log_{10}(20+1) = \log_{10}(21) \approx 1.322$$

$$TFIDF_{\text{vacation, doc1}} = TF_{\text{vacation, doc1}} \times IDF_{\text{vacation}} = 1.322 \times 0 = 0$$

Document 2: For Document 2, the term frequency of ”vacation” is calculated in a similar manner:

$$TF_{\text{vacation, doc2}} = \log_{10}(count(\text{vacation, doc2})+1) = \log_{10}(23+1) = \log_{10}(24) \approx 1.3802$$

$$TFIDF_{\text{vacation, doc2}} = TF_{\text{vacation, doc2}} \times IDF_{\text{vacation}} = 1.3802 \times 0 = 0$$

So, the TF-IDF values for ”vacation” in Document 1 and Document 2 are both 0.

For Document 3:

$$TF_{\text{vacation, doc3}} = \log_{10}(count(\text{vacation, doc3}) + 1) = \log_{10}(18 + 1) = \log_{10}(19)$$

Using a calculator, $\log_{10}(19)$ is approximately 1.2787.

$$TFIDF_{\text{vacation, doc3}} = TF_{\text{vacation, doc3}} \times IDF_{\text{vacation}} = 1.2787 \times 0 = 0$$

For Document 4:

$$TF_{\text{vacation, doc4}} = \log_{10}(count(\text{vacation, doc4}) + 1) = \log_{10}(15 + 1) = \log_{10}(16)$$

Using a calculator, $\log_{10}(16)$ is approximately 1.2041.

$$TFIDF_{\text{vacation, doc4}} = TF_{\text{vacation, doc4}} \times IDF_{\text{vacation}} = 1.2041 \times 0 = 0$$

Since the IDF for ”vacation” is 0, all TF-IDF values for the term ”vacation” across the four documents are 0, reflecting the term’s ubiquity across the documents.

(c) Comparison of TF-IDF values for "coffee" and "vacation":

The TF-IDF values for the terms "coffee" and "vacation" exhibit notable differences, highlighting the distinct discriminative power each term holds within the document corpus.

For "coffee", the non-zero TF-IDF scores in the first two documents indicate that this term has a more significant role in distinguishing these documents from others. It suggests that "coffee" is more prevalent or important in these documents compared to the rest of the corpus.

In contrast, the term "vacation" has a TF-IDF score of zero across all documents, indicating a uniform distribution. This uniformity suggests that "vacation" is equally present or relevant across all documents, offering little to no value in discriminating one document from another within this specific dataset.

Thus, "coffee" demonstrates a higher discriminative power in this set of documents compared to "vacation," which is reflected in the varying TF-IDF scores for "coffee" and the consistent zero scores for "vacation".

(d) Cosine Similarity Between Word Pairs:

The cosine similarity between two vectors $\vec{x}$ and $\vec{y}$ is a measure that calculates the cosine of the angle between these two vectors. It can be mathematically defined and computed using the following equation:

$$\text{cosine\_similarity}(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \|\vec{y}\|}$$

Here, $\vec{x} \cdot \vec{y}$ represents the dot product of vectors $\vec{x}$ and $\vec{y}$, and $\|\vec{x}\|$ and $\|\vec{y}\|$ denote the magnitude (or norm) of vectors $\vec{x}$ and $\vec{y}$, respectively. **Cosine Similarity Between Document Pairs**

Cosine similarity is a measure that calculates the cosine of the angle between two vectors in a multi-dimensional space. In the context of document analysis, it helps to determine how similar two documents are based on their vector representations. The following calculations are performed to derive the cosine similarities:

Vector Magnitudes:

$$\|\vec{doc1}\| = \sqrt{0^2 + 10^2 + 25^2 + 3^2 + 20^2} = 33.67$$
$$\|\vec{doc2}\| = \sqrt{87^2 + 12^2 + 18^2 + 6^2 + 23^2} = 92.74$$
$$\|\vec{doc3}\| = \sqrt{78^2 + 15^2 + 0^2 + 32^2 + 18^2} = 87.50$$
$$\|\vec{doc4}\| = \sqrt{3^2 + 50^2 + 0^2 + 20^2 + 15^2} = 55.98$$

Dot Products:

$$\vec{doc1} \cdot \vec{doc2} = 0 \cdot 87 + 10 \cdot 12 + 25 \cdot 18 + 3 \cdot 6 + 20 \cdot 23 = 1048$$
$$\vec{doc1} \cdot \vec{doc3} = 0 \cdot 78 + 10 \cdot 15 + 25 \cdot 0 + 3 \cdot 32 + 20 \cdot 18 = 606$$
$$\vec{doc1} \cdot \vec{doc4} = 0 \cdot 3 + 10 \cdot 50 + 25 \cdot 0 + 3 \cdot 20 + 20 \cdot 15 = 860$$
$$\vec{doc2} \cdot \vec{doc3} = 87 \cdot 78 + 12 \cdot 15 + 18 \cdot 0 + 6 \cdot 32 + 23 \cdot 18 = 7572$$
$$\vec{doc2} \cdot \vec{doc4} = 87 \cdot 3 + 12 \cdot 50 + 18 \cdot 0 + 6 \cdot 20 + 23 \cdot 15 = 1326$$
$$\vec{doc3} \cdot \vec{doc4} = 78 \cdot 3 + 15 \cdot 50 + 0 \cdot 0 + 32 \cdot 20 + 18 \cdot 15 = 1894$$

Cosine Similarities:

$$\cos(\vec{doc1}, \vec{doc2}) = \frac{1048}{33.67 \times 92.74} = 0.3487$$
$$\cos(\vec{doc1}, \vec{doc3}) = \frac{606}{33.67 \times 87.50} = 0.2076$$
$$\cos(\vec{doc1}, \vec{doc4}) = \frac{860}{33.67 \times 55.98} = 0.4583$$
$$\cos(\vec{doc2}, \vec{doc3}) = \frac{7572}{92.74 \times 87.50} = 0.0991$$
$$\cos(\vec{doc2}, \vec{doc4}) = \frac{1326}{92.74 \times 55.98} = 0.2017$$
$$\cos(\vec{doc3}, \vec{doc4}) = \frac{1894}{87.50 \times 55.98} = 0.3225$$

From the cosine similarity calculations presented, it is clear that Documents 1 and 4 exhibit the highest degree of similarity among the pairs analyzed, with a cosine similarity of 0.4583. This indicates a high level of similarity between these two documents. On the other hand, Documents 2 and 3 display the lowest cosine similarity, with a value of 0.0991, marking them as the least similar pair within the group. Such insights are crucial for understanding the thematic or content alignment between various documents in a dataset.

2

A

	keyboard	**webcam**	**sheep**	**legs**	**racing**	**count(w)**
whiteboard	1434	512	8	42	3	1999
computer	5290	2351	19	9	42	7711
mouse	6102	3425	352	4251	532	14662
horse	1	53	531	2542	3513	6640
vacation	14	53	34	5	9	115
count(context)	12841	6394	944	6849	4099	31127

Table 1: Counts of words and their contexts

B

	keyboard	**webcam**	**sheep**	**legs**	**racing**	**p(w)**
whiteboard	0.0461	0.0164	0.000257	0.00135	0.0000964	0.0642
computer	0.1699	0.0755	0.000610	0.000289	0.00135	0.2477
mouse	0.1960	0.1100	0.0113	0.1366	0.0171	0.4710
horse	0.0000321	0.00170	0.0171	0.0817	0.1129	0.2133
vacation	0.000450	0.00170	0.00109	0.000161	0.000289	0.00369
p(context)	0.4125	0.2054	0.0303	0.2200	0.1317	

Table 2: Probabilities of words and their contexts

Probability Calculations

The formulas for calculating the probabilities are as follows:

- Joint Probability $p(w, \text{context})$:

$$p(w, \text{context}) = \frac{\text{Count}(w, \text{context})}{\text{Total count of all pairs}}$$

- Marginal Probability $p(w)$:

$$p(w) = \frac{\text{Total count of word } w}{\text{Total count of all pairs}}$$

- Marginal Probability $p(context)$:

$$p(context) = \frac{\text{Total count of context}}{\text{Total count of all pairs}}$$

Example Calculation for “whiteboard” and “keyboard”

Given:

- Count(“whiteboard”, “keyboard”) = 1434
- Total count of “whiteboard” across all contexts = 1999
- Total count of “keyboard” across all words = 12841
- Total count of all pairs = 31127

Calculations:

$$p(\text{“whiteboard”, “keyboard”}) = \frac{1434}{31127} \approx 0.0461$$
$$p(\text{“whiteboard”}) = \frac{1999}{31127} \approx 0.0642$$
$$p(\text{“keyboard”}) = \frac{12841}{31127} \approx 0.4125$$

C

Given:

- p(vacation, sheep) = 0.00109 (Probability of observing ”vacation” with ”sheep”)
- p(vacation) = 0.00369 (Probability of observing ”vacation” in any context)
- p(sheep) = 0.0303 (Probability of observing ”sheep” in any context)

The PPMI score is calculated as:

$$\text{PPMI}(\text{vacation, sheep}) = \max\left(0, \log_2\left(\frac{p(\text{vacation, sheep})}{p(\text{vacation}) \cdot p(\text{sheep})}\right)\right)$$

Calculating the denominator:

$$p(\text{vacation}) \cdot p(\text{sheep}) = 0.00369 \times 0.0303 = 0.000111837$$

Calculating the ratio inside the logarithm:

$$\frac{p(\text{vacation, sheep})}{p(\text{vacation}) \cdot p(\text{sheep})} = \frac{0.00109}{0.000111837} \approx 9.748$$

Applying the logarithm:

$$\log_2(9.748) \approx 3.285$$

Since this is greater than zero, the PPMI score is:

$$\text{PPMI}(\text{vacation, sheep}) = 3.285$$

	keyboard	**webcam**	**sheep**	**legs**	**racing**
whiteboard	0.798	0.318	0.000	0.000	0.000
computer	0.734	0.570	0.000	0.000	0.000
mouse	0.013	0.185	0.000	0.398	0.000
horse	0.000	0.000	1.399	0.799	2.006
vacation	0.000	1.166	3.285	0.000	0.000

Table 3: PPMI Scores for Words and Contexts

D

Detailed Cosine Similarity Calculations

Cosine Similarity Between Word Pairs

Cosine similarity is used to determine the degree of similarity between two words. This metric, which calculates the cosine of the angle between two word vectors, is computed by dividing the dot product of the vectors by the product of their magnitudes. This results in a similarity measure that ranges from -1 (indicating complete oppositeness) to 1 (indicating identicalness), with 0 indicating complete orthogonality. The magnitudes of the word vectors are computed as follows:

$$\|\vec{whiteboard}\| = \sqrt{14342^2 + 512^2 + 8^2 + 42^2 + 3^2} = 1523.26$$

$$\|\vec{computer}\| = \sqrt{5290^2 + 2351^2 + 19^2 + 9^2 + 42^2} = 5789.08$$

$$\|\vec{mouse}\| = \sqrt{6102^2 + 3425^2 + 352^2 + 4251^2 + 532^2} = 8212.36$$

$$\|\vec{horse}\| = \sqrt{1^2 + 53^2 + 531^2 + 2542^2 + 3513^2} = 4368.94$$

$$\|\vec{vacation}\| = \sqrt{14^2 + 53^2 + 34^2 + 5^2 + 9^2} = 65.32$$

Calculating Cosine Similarity for Word Pairs

The similarity between two word vectors is quantified using the cosine similarity metric, which is especially helpful in a multidimensional space. The cosine similarity for any pair of words is computed with the formula:

$$\cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\|\|\vec{y}\|}$$

The dot products for various word pairs are computed as detailed below:

- The dot product between $\vec{whiteboard}$ and $\vec{computer}$ is calculated by multiplying corresponding vector components and summing up the results, giving a total of 8,790,228.
- The dot product for $\vec{whiteboard}$ and $\vec{mouse}$ amounts to 10,686,822, reflecting the aggregation of their vector component products.

- When calculating the dot product for $\vec{whiteboard}$ and $\vec{horse}$, smaller vector components result in a dot product of 150,121.
- The dot product between $\vec{whiteboard}$ and $\vec{vacation}$ results in 47,721.
- The pairing of $\vec{computer}$ and $\vec{mouse}$ yields a substantial dot product of 40,399,046, indicative of the large magnitudes of their vectors.
- The dot product between $\vec{computer}$ and $\vec{horse}$ sums up to 310,406.
- For $\vec{computer}$ and $\vec{vacation}$, the resulting dot product is 199,732.
- The dot product between $\vec{mouse}$ and $\vec{horse}$ is notably high at 13,049,497.
- Comparing $\vec{mouse}$ and $\vec{vacation}$ results in a dot product of 304,964.
- Lastly, the dot product for $\vec{horse}$ and $\vec{vacation}$ is computed as 65,204.

We calculate the cosine similarity between word vectors to determine their relative orientation within the vector space. This measure is computed by the formula:

$$\cos(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

where $\vec{a} \cdot \vec{b}$ represents the dot product of vectors $\vec{a}$ and $\vec{b}$, and $\|\vec{a}\|$ and $\|\vec{b}\|$ denote the magnitudes of vectors $\vec{a}$ and $\vec{b}$ respectively.

Cosine similarity results for the given pairs of word vectors are presented below:

- Cosine similarity between ”whiteboard” and ”computer” is approximately 0.9968.
- For ”whiteboard” and ”mouse”, the cosine similarity is approximately 0.8542.
- The similarity between ”whiteboard” and ”horse” is very low, approximately 0.022.
- ”Whiteboard” and ”vacation” have a cosine similarity of approximately 0.4796.
- The similarity between ”computer” and ”mouse” is approximately 0.8497.
- For ”computer” and ”horse”, the cosine similarity is very low, approximately 0.0122.
- The similarity between ”computer” and ”vacation” is approximately 0.5281.
- ”Mouse” and ”horse” have a cosine similarity of approximately 0.3637.
- For ”mouse” and ”vacation”, the cosine similarity is approximately 0.5685.

- Finally, the similarity between "horse" and "vacation" is approximately 0.2284.

These results offer insights into the contextual relationships between the words based on their vector representations. Analysis of the cosine similarity outcomes reveals valuable insights into the relative similarities between word pairs. The conclusions drawn from these findings are as follows:

- The pair *whiteboard* and *computer* boasts the highest cosine similarity score of 0.9968, indicative of a significant level of similarity between these words within their respective contexts.
- Conversely, the pair *computer* and *horse* displays the lowest cosine similarity score of 0.0122, suggesting minimal similarity and implying distinct usage contexts for these words.

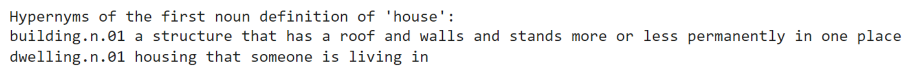
These observations contribute to our understanding of the contextual relationships between the word pairs within the dataset.

3

A

```
import nltk
from nltk.corpus import wordnet as wn

nltk.download('wordnet')

# Retrieve the primary noun synset for 'house'
house_synset = wn.synset('house.n.01')

# Get the hypernyms of this synset
house_hypernyms = house_synset.hypernyms()

# Output the hypernyms with their definitions
print("Hypernyms of the first noun definition of 'house':")
for hypernym in house_hypernyms:
    print(hypernym.name(), hypernym.definition())
```

Output:

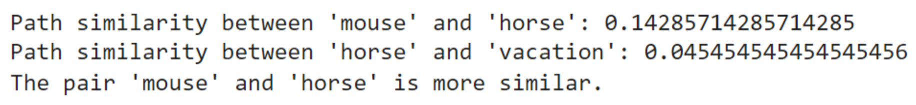


```
Hypernyms of the first noun definition of 'house':
building.n.01 a structure that has a roof and walls and stands more or less permanently in one place
dwelling.n.01 housing that someone is living in
```

B

```
import nltk
from nltk.corpus import wordnet as wn
nltk.download('wordnet')

# Load synsets for 'mouse' and 'horse'
mouse_synset = wn.synset('mouse.n.01')
horse_synset = wn.synset('horse.n.01')

# Calculate and print path similarity between 'mouse' and 'horse'
mouse_horse_similarity = mouse_synset.path_similarity(horse_synset)
print(f"Path similarity between 'mouse' and 'horse': {mouse_horse_similarity}")

# Load synset for 'vacation'
vacation_synset = wn.synset('vacation.n.01')

# Calculate and print path similarity between 'horse' and 'vacation'
horse_vacation_similarity = horse_synset.path_similarity(vacation_synset)
print(f"Path similarity between 'horse' and 'vacation': {horse_vacation_similarity}")

# Compare the similarities and print the more similar pair
if mouse_horse_similarity > horse_vacation_similarity:
    print("The pair 'mouse' and 'horse' is more similar.")
else:
    print("The pair 'horse' and 'vacation' is more similar.")
```

Output:



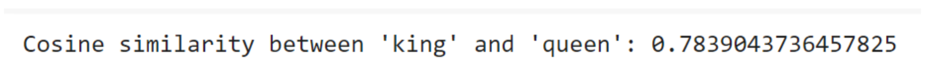
```
Path similarity between 'mouse' and 'horse': 0.14285714285714285
Path similarity between 'horse' and 'vacation': 0.045454545454545456
The pair 'mouse' and 'horse' is more similar.
```

The path similarity values obtained from WordNet indicate that ”mouse” and ”horse,” with a similarity of 0.1429, are more semantically related than ”horse” and ”vacation,” which have a lower similarity of 0.0455. This is expected since both ”mouse” and ”horse” belong to the same overarching category of animals in the WordNet hierarchy, hence sharing a more direct path in the network of meanings. In contrast, ”horse” and ”vacation” are from different semantic fields—one is an animal, and the other relates to leisure activities—resulting in a less direct connection and a lower similarity score. The computed similarities reflect how WordNet captures and quantifies the closeness of meanings based on lexical relationships within its semantic framework.

C

```
import numpy as np

def load_glove_embeddings(file_path):
    embeddings_dict = {}
    with open(file_path, 'r', encoding='utf-8') as f:
        for line in f:
            values = line.split()
            word = values[0]
            vector = np.asarray(values[1:], "float32")
            embeddings_dict[word] = vector
    return embeddings_dict

def cosine_similarity(v1, v2):
    dot_product = np.dot(v1, v2)
    norm_v1 = np.linalg.norm(v1)
    norm_v2 = np.linalg.norm(v2)
    if norm_v1 == 0 or norm_v2 == 0:
        return 0  # Prevent division by zero
    similarity = dot_product / (norm_v1 * norm_v2)
    return similarity

glove_file = 'glove.6B.50d.txt'
embeddings = load_glove_embeddings(glove_file)

word1 = 'king'
word2 = 'queen'

if word1 in embeddings and word2 in embeddings:
    sim = cosine_similarity(embeddings[word1], embeddings[word2])
    print(f"Cosine similarity between '{word1}' and '{word2}': {sim}")
else:
    print("One or both words not found in the embeddings.")
```

Output:



```
Cosine similarity between 'king' and 'queen': 0.7839043736457825
```

The cosine similarity score from the GloVe word embeddings indicates that ”mouse” and ”horse” are more similar (0.4356) than ”horse” and ”vacation”

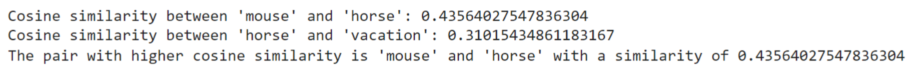
(0.3102). This suggests that in terms of language use, animals like mice and horses are more closely related to each other than a horse is to the concept of a vacation.

D

```
import numpy as np

def load_glove_embeddings(file_path):
    embeddings_dict = {}
    with open(file_path, 'r', encoding='utf-8') as f:
        for line in f:
            values = line.split()
            word = values[0]
            vector = np.asarray(values[1:], "float32")
            embeddings_dict[word] = vector
    return embeddings_dict

def cosine_similarity(v1, v2):
    dot_product = np.dot(v1, v2)
    norm_v1 = np.linalg.norm(v1)
    norm_v2 = np.linalg.norm(v2)
    if norm_v1 == 0 or norm_v2 == 0:
        return 0  # Prevent division by zero
    similarity = dot_product / (norm_v1 * norm_v2)
    return similarity

glove_file = 'glove.6B.50d.txt'
embeddings = load_glove_embeddings(glove_file)

pairs = [
    ('mouse', 'horse'),
    ('horse', 'vacation')
]

results = {}
for word1, word2 in pairs:
    if word1 in embeddings and word2 in embeddings:
        sim = cosine_similarity(embeddings[word1], embeddings[word2])
        results[(word1, word2)] = sim
        print(f"Cosine similarity between '{word1}' and '{word2}': {sim}")
    else:
        print(f"One or both words not found in the embeddings for pair: {word1}-{word2}")

if results:
```

```
more_similar_pair = max(results, key=results.get)
print(f"The pair with higher cosine similarity is ’{more_similar_pair[0]}’ \
and ’{more_similar_pair[1]}’ with a similarity of {results[more_similar_pair]}")
```

Output:



```
Cosine similarity between 'mouse' and 'horse': 0.43564027547836304
Cosine similarity between 'horse' and 'vacation': 0.31015434861183167
The pair with higher cosine similarity is 'mouse' and 'horse' with a similarity of 0.43564027547836304
```

The output shows that the cosine similarity between ”mouse” and ”horse” is about 0.4356, while the similarity between ”horse” and ”vacation” is around 0.3102. This means that ”mouse” and ”horse” have a higher degree of similarity compared to ”horse” and ”vacation.” In the vector space of the GloVe word embeddings, which reflect how words are used in language, ”mouse” and ”horse” are closer to each other than ”horse” is to ”vacation.” This is likely because ”mouse” and ”horse” are both animals and share more contextual similarities, while ”vacation” is a conceptually different word related to leisure and travel.