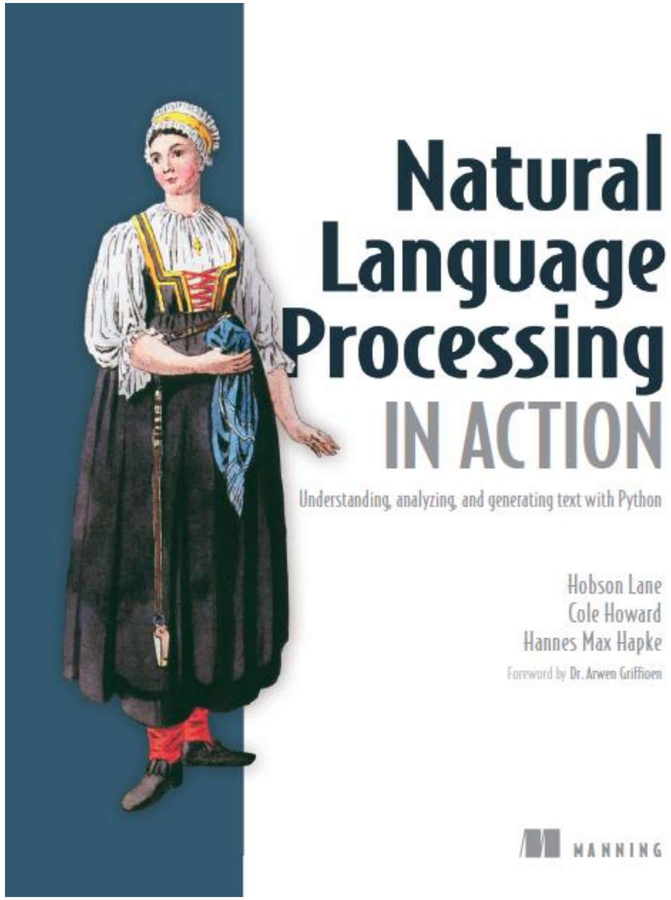
Math with Words

Prof. Gheith Abandah

Reference 1

- Chapter 3: **Math with words (TF-IDF vectors)**



- H. Lane, C. Howard, and H. Hapke, **Natural Language Processing in Action:** Understanding, analyzing, and generating text with Python, Manning, 2019.

Outline

- Types of Bags of Words
- Zipf's Law
- Term Frequency (TF)
- Inverse Document Frequency (IDF)
- TF-IDF Vectors
- Cosine Similarity
- Summary

Types of Bags of Words

Definition: A simple NLP model representing text data based on word occurrence.

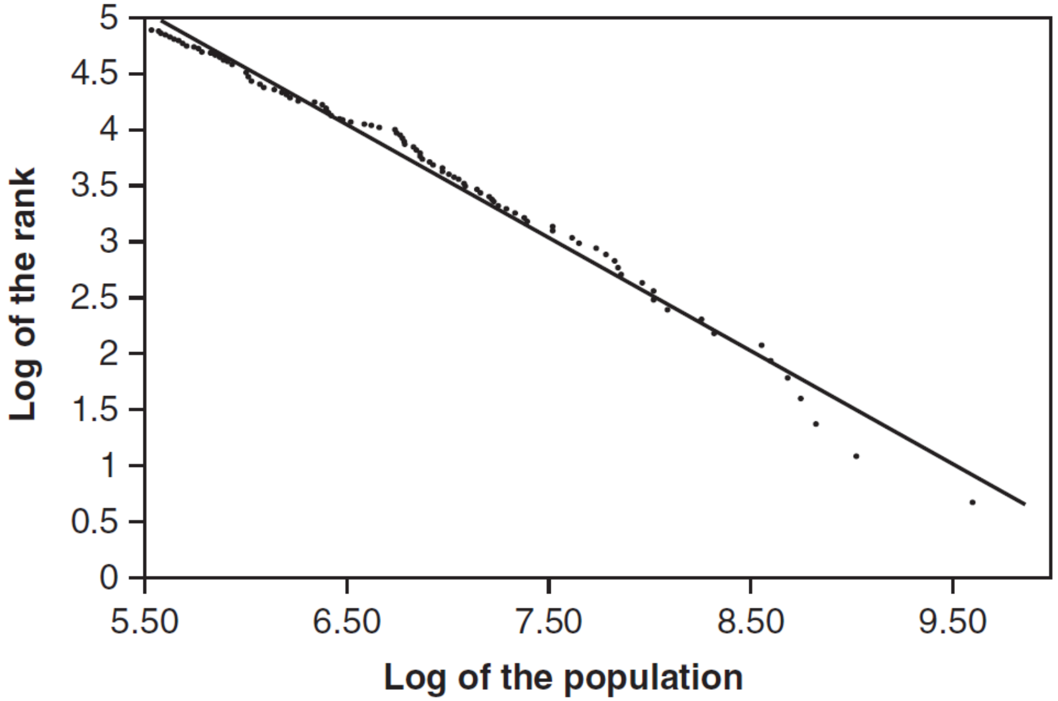
1. **Standard Bag of Words**: Counts word occurrences in a document.
2. **Binary Bag of Words**: Indicates presence (1) or absence (0) of words.
3. **N-Grams**: Extends BoW by including word pairs or tuples as features.
4. **TF (Term Frequency)**: Adjusts BoW counts to reflect frequency rather than occurrence.

Zipf's Law in Natural Languages

- **Definition**: Observes that the frequency of any word is inversely proportional to its rank in the frequency table.
- **Implication**: A few words are used very often, while many are used rarely.
- **Example**: "the", "is", and "and" often appear at the top of English word frequency counts.
- **Application**: Helps in understanding natural language patterns and optimizing algorithms.

Zipf's Law Examples

City Population Distribution



Most common words in a document

```
>>> token_counts.most_common(20)
[('the', 69971),
 ('of', 36412),
 ('and', 28853),
 ('to', 26158),
 ('a', 23195),
 ('in', 21337),
 ('that', 10594),
 ('is', 10109),
 ('was', 9815),
 ('he', 9548),
 ('for', 9489),
 ('it', 8760),
 ('with', 7289),
 ('as', 7253),
 ('his', 6996),
 ('on', 6741),
 ('be', 6377),
 ('at', 5372),
 ('by', 5306),
 ('i', 5164)]
```

Term Frequency (TF)

- **Definition**: Measures how frequently a term appears in a document.
- TF(t, d) = (Number of times term t appears in document d) ÷ (Total number of terms in document d)
- **Purpose**: To normalize word counts based on document length.

Example

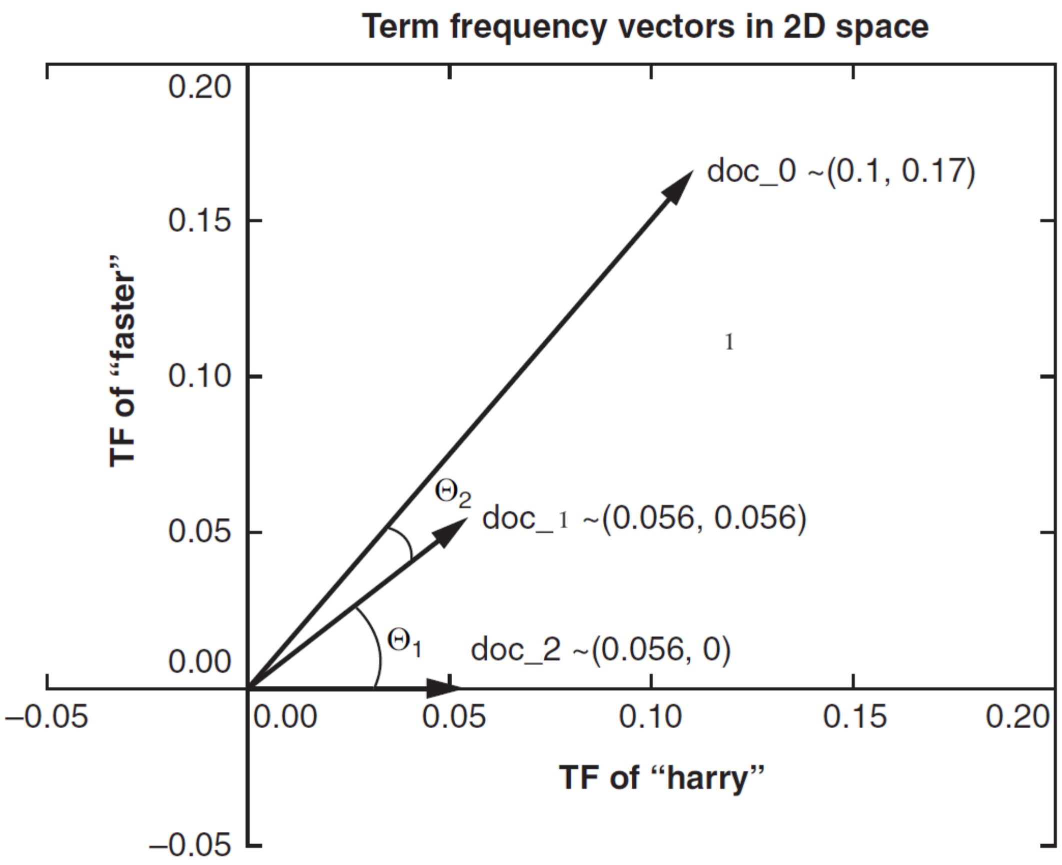
```python
from collections import Counter

def compute_tf(document):
    words = document.lower().split()
    word_counts = Counter(words)
    total_words = len(word_counts)
    tf = {word: count / total_words for word, count in
          word_counts.items()}
    return tf
# Example document
document = "Cats love playing with cats"
tf = compute_tf(document)
print(tf)
```

```
{'cats': 0.5, 'love': 0.25, 'playing': 0.25, 'with': 0.25}
```

Representing the Words of a Document as a Vector

- **Vector Space Model**: Conceptualizes documents as vectors of identifiers.
- **Dimensions**: Each unique word represents a dimension in the vector space.
- **Word Vectors**: Documents are encoded as numerical vectors based on word occurrences or TF.
- **Use**: Facilitates the comparison of documents through mathematical operations.



Inverse Document Frequency (IDF)

- **Definition**: Measures how important a term is across a set of documents.
- IDF(t, D) = log(Total number of documents D ÷ Number of documents with term t)
- **Purpose**: To weigh down the frequent terms while scaling up the rare ones.

Example

```
import math
# Total number of documents
N = 100000
# Number of documents containing each term
n_apple = 20000
n_quantum = 500

# Calculate IDF for each term
idf_apple = math.log(N / n_apple)
idf_quantum = math.log(N / n_quantum)

print(idf_apple, idf_quantum)

1.6094379124341003 5.298317366548036
```

TF-IDF Vectors

- **Combination**: Multiplication of TF and IDF scores for each term.
- TF-IDF(t, d, D) = TF(t, d) × IDF(t, D)

- **Result**: Reflects the importance of words within documents relative to a document set.

Applications of TF-IDF

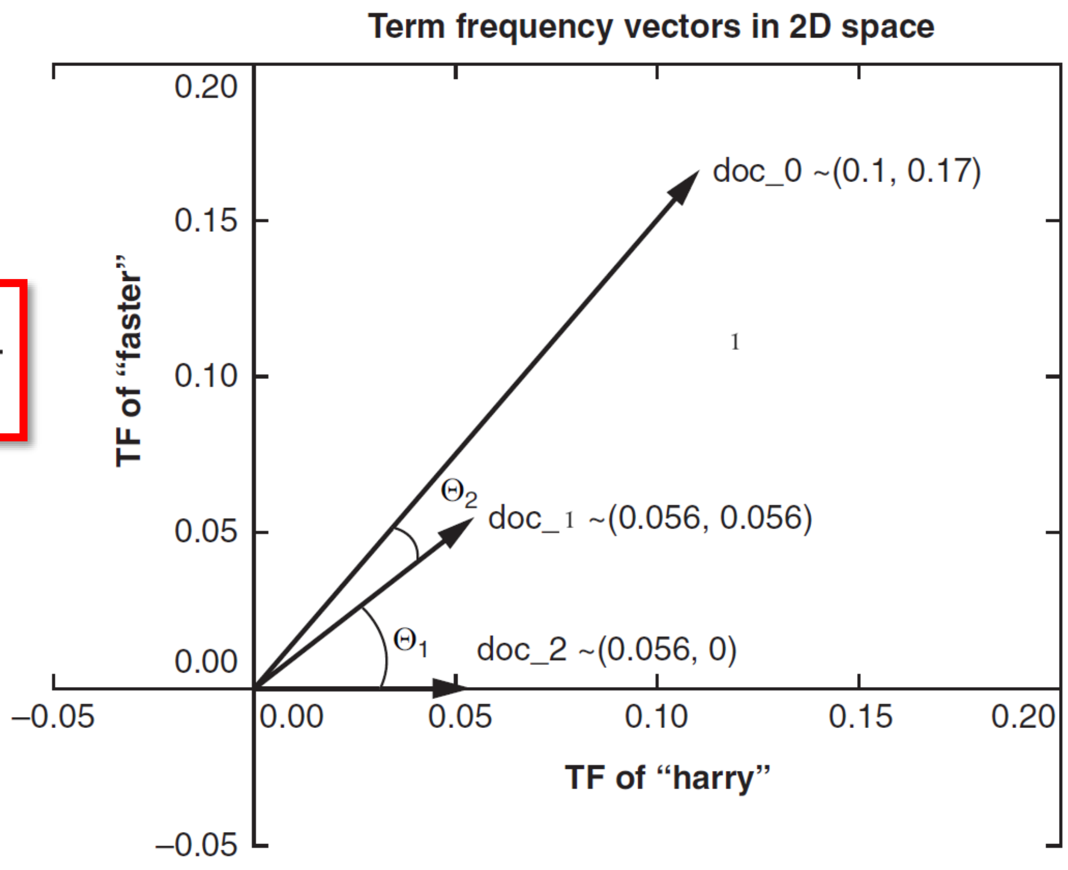
- **Information Retrieval**: Improves search engine relevance by scoring document relevance.
- **Document Classification**: Helps in categorizing documents into different classes.
- **Feature Selection**: Identifies relevant words for use in machine learning models.
- **Text Summarization**: Assists in identifying key sentences based on term significance.

Estimating Similarity Using Cosine Similarity

- **Definition**: Measures the cosine of the angle between two vectors.
- **Application**: Determines the similarity between two documents in the vector space.
- Cosine Similarity = (Dot product of vectors) ÷ (Product of their magnitudes)

$$\boxed{\cos\Theta = \frac{A \cdot B}{|A|\,|B|}}$$

- **Range**: -1 (opposite) to 1 (identical), where 0 indicates orthogonality (no similarity).



Example: Similarity using TF-IDF Vectors

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

# Example documents
document_1 = "The school is large."
document_2 = "His school is my school too."
document_3 = "The student goes to school."

# Put the documents into a list for vectorization
documents = [document_1, document_2, document_3]

# Create a TfidfVectorizer object
tfidf_vectorizer = TfidfVectorizer()

# Fit and transform the documents
tfidf_matrix = tfidf_vectorizer.fit_transform(documents)
```

Example: Similarity using TF-IDF Vectors

```
print(tfidf_vectorizer.get_feature_names_out())
print(tfidf_matrix.shape)

# Calculate the cosine similarity between the three documents
similarity = cosine_similarity(tfidf_matrix[0:3],
                               tfidf_matrix[0:3])

print(f"Cosine Similarity: {similarity}")
```

```
['goes' 'his' 'is' 'large' 'my' 'school' 'student' 'the' 'to'
 'too']
(3, 10)
Cosine Similarity: [[1.         0.36146878 0.29558668]
                    [0.36146878 1.         0.15785465]
                    [0.29558668 0.15785465 1.        ]]
```

Summary

- Types of Bags of Words
- Zipf’s Law
- Term Frequency (TF)
- Inverse Document Frequency (IDF)
- TF-IDF Vectors
- Cosine Similarity