Introduction to Natural Language Processing

BUS 243 F: Spring 2023

Yeabin Moon

Lecture 3

Meaning of relative counts

- Bag-of-words representation has a lot of interesting applications
- Now we are going to consider the counts in relation to the rest of the documents
- Careful on the readings
  - Note that the TF-IDF representation is not an improvement from the BOW representation
    - They are different
  - Also, it is not certain that whether TF-IDF represents the meaning of words

Popular representation?

- TF-IDF stands for **t**erm **f**requency times **i**nverse **d**ocument **f**requency
  - Term frequencies are the counts of each word in a document
  - Inverse document frequency means that you’ll divide each of those word counts by the number of documents in which the word occurs
- Popular representation of the text
  - Not much popular in the front end
  - Somewhat overrated, need to have a discussion

Assumption

- Vector representation of the text based on the frequency of each word in the given text
- Assume that the more times a word occurs, the more meaning it must contribute to that document
  - Each word has some innate sentiment
  - Idea of text classification based on BOW representation
- The main question is whether we need to normalize the counts
- Let’s see the first example code

0.1818?

- What does this number mean?
- Sentence: *The faster Harry got to the store, the faster Harry, the faster, would get home.*
  - `Harry' used twice (after case folding)
  - The number of unique tokens is 11
- Not clear why do we ever need the normalization
- Worse yet, why do we need the number of unique tokens as the normalizer (denominator)?
- Not exactly sure of the meaning of 0.1818

Textbook example: why “temper” it all?

- TF(“dog,” document A) = 3/30 = .1
- TF(“dog,” document B) = 100/580000 = .00017
- What’s the meaning of TF(“dog,” document A), 0.1?
- I see the reasoning behind the normalization
- But still not clear why do we need the unique tokens in the previous example
- Am I too picky?

What are we missing?

- For a long text, the normalized count of a word would mean more
  - The Dog example would make sense
- Not just conceptually, the calculation result (term frequency) would be robust
  - Normalizer: the number of tokens used vs. the number of unique tokens used
    - Strong assumption?
- However, for a short text, each approach would result in very different outcomes
  - Think about tokenizers
- For most NLP applications, there would be no definitive formula
  - More important thing is to understand the concepts
- Let’s see Kite example in the code

What are we doing?

- document_vectors is the python list
  - Each item is the Ordered dictionary
  - Data type is in fact not very important here
  - Let’s interpret each item
- What’s the denominator?
  - Does it make sense to you?
  - Why not using the number of unique tokens in the corresponding sentence?
  - Why not using the number of every tokens?

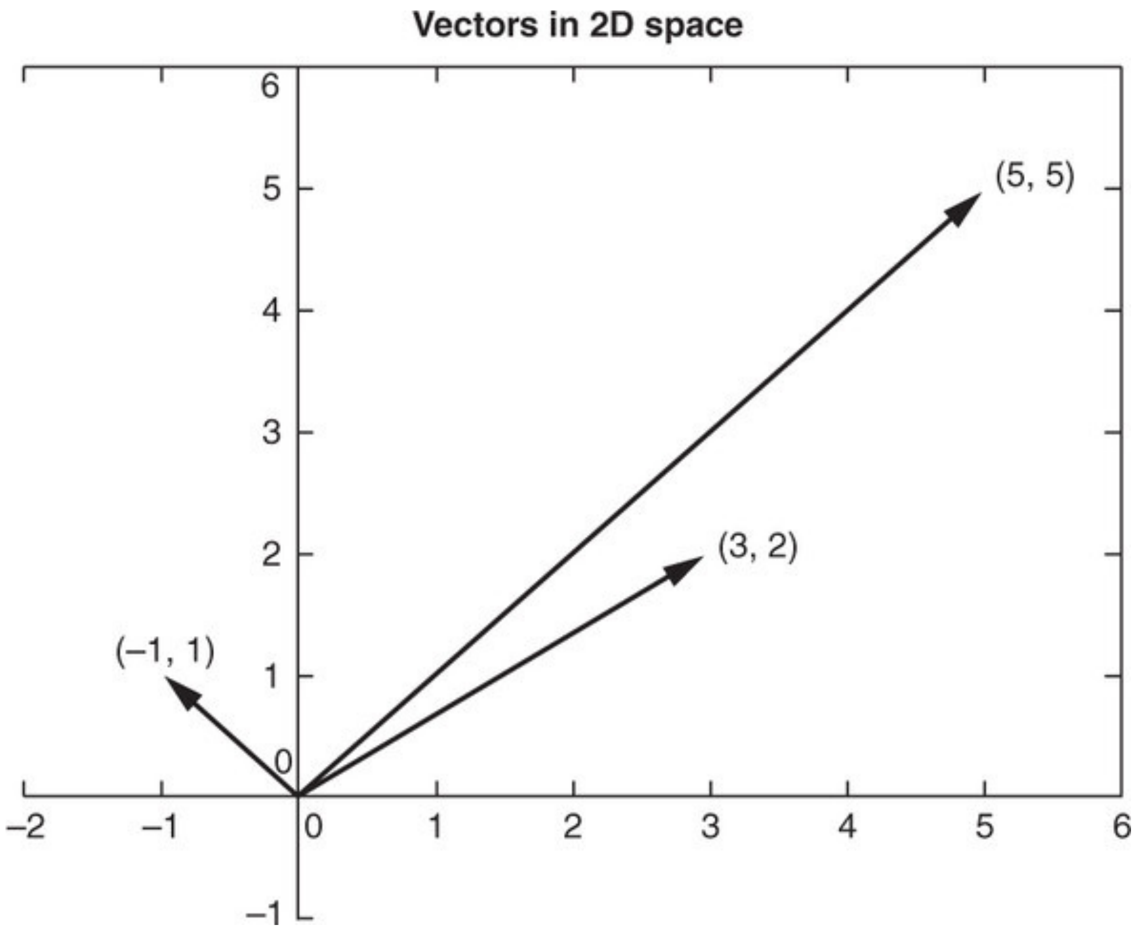
Understanding the construction

- Each of your three document vectors will need to have 18 values, even if the document for that vector doesn’t contain all 18 words in your lexicon
- Each token is assigned a *slot* in your vectors corresponding to its position in your lexicon
- You have three vectors, one for each document. So what? What can you do with them?

Vector spaces

- Vectors are the primary building blocks of linear algebra, or vector algebra
  - Python Tuples
  - A tuple is a collection which is ordered and unchangeable
- Order means there is a particular direction of a vector and possible to measure the distance between vectors
- What’s the meaning of dimensionality?

What can we tell about the following 3 vecs?

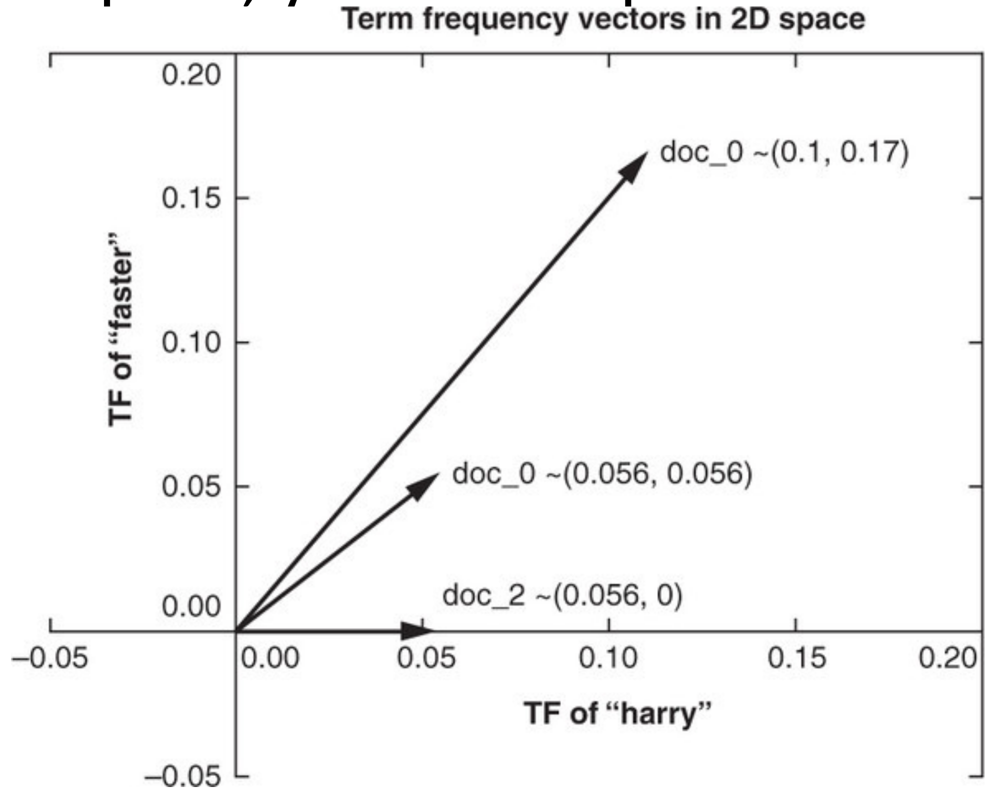


Dimensionality

- More dimensionality means more information
- The linear algebra all works out the same!
- You might need more computing power as the dimensionality grows
  - Curse of dimensionality issues
- For a natural language document vector space, the dimensionality of your vector space is the count of the number of distinct words that appear in the entire corpus
- You can then describe each document within this K-dimensional vector space by a K-dimensional vector

Vector algebra

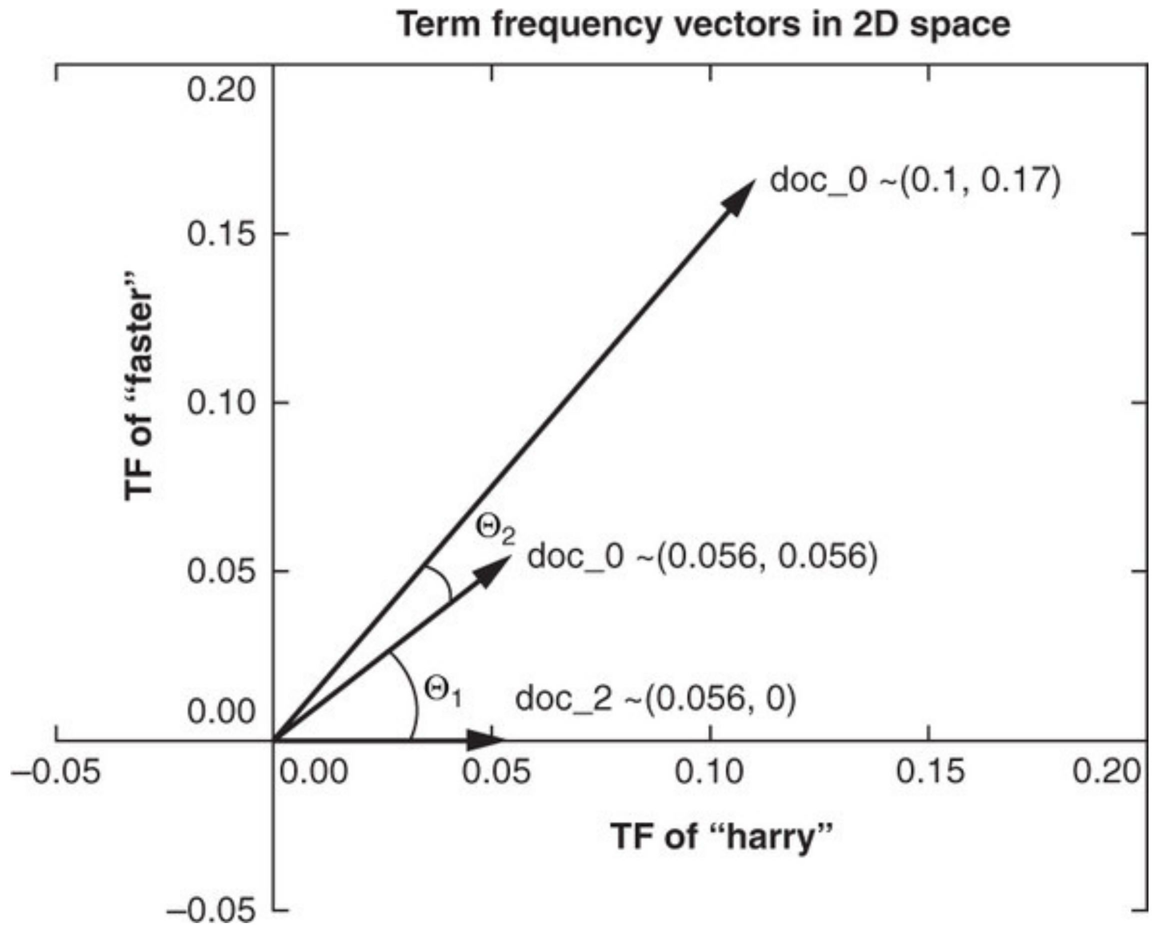
- Now that you have a representation of each document and know they share a common space, you have a path to compare them



Similarity

- Two vectors are *similar* if they share similar direction
  - What’s the meaning of longer vector?
- You’d like your estimate of document similarity to find use of the same words about the same number of times in similar proportions
- This accurate estimate would give you confidence that the documents they represent are probably talking about similar things
  - You’re right. Keep your doubt aside for now
- $\cos\theta = \frac{A \cdot B}{|A||B|}$

Cosine Similarity



Cosine representation

- The range of cosine values is between -1 and 1
  - It’s the cosine of the angle between these two vectors
  - It gives you a value for how much the vectors point in the same direction
- $\cos\theta = 1$ represents the documents are using similar words in similar proportion
  - The documents whose document vectors are close to each other are likely talking about the same thing
- $\cos\theta = 0$ represents two vectors that share no components
  - For NLP TF vectors, this situation occurs only if the two documents share no words in common

Assumptions?

- What do we need to assume to conclude the meaning of cosine values?
  - They just use either completely same (different) words
  - What about synonyms and antonyms?
  - Hard to infer semantics or meanings in general
- A cosine similarity of -1 represents two vectors that are anti-similar, completely opposite
  - Impossible for TF since every component should be non-negative values

Zipf’s Law

… the frequency of occurrence of any given word in a language is inversely proportional to its rank in the frequency table. In other words, the most frequently occurring word in a language (such as "the" in English) will occur approximately twice as often as the second most frequent word (such as "of"), three times as often as the third most frequent word, and so on…

ChatGPT

Application

- *Inverse proportionality* refers to a situation where an item in a ranked list will appear with a frequency tied explicitly to its rank in the list
    - If you see any outliers that don’t fall along a straight line in a log-log plot, it may be worth investigating
- Let’s take look the code on the relationship between the population of US cities and the rank of that population and the Brown corpus

Ok, so what does it describe?

- The distribution over words (and other linguistic elements) resembles that of a **power law**
  - There will be a few words that are very frequent, and a long tail of words that are rare
- A consequence is that NLP algorithms must be especially robust to observations that do not occur in the training data

The meaning of Normalized count

- What’s the idea behind of power law?
  - What’s the assumption of the count-base representation?
  - Now we consider a relative count in document vectors
- Inverse document frequency, or IDF, is your window through Zipf law
- Let’s go back to the Wikipedia document regarding a kite

What are we doing?

- Term Frequency of "kite" in intro is: 0.0441
- Term Frequency of "kite" in history is: 0.0202
- Term Frequency of "and" in intro is: 0.0275
- Term Frequency of "and" in history is: 0.0303

Text data structure

- Meaning unit of text
  - Token(s) → Sentence(s) → Document(s) → Corpus (Corpora)
- But it could be
  - Token(s) → Corpus (Corpora)
  - Token(s) → Sentence(s) → Corpus (Corpora)
  - Token(s) → Documents(s) → Corpus (Corpora)
- Tokens are the basic units of meaning in a text
  - Words, numbers, punctuation marks,…
- A corpus consists of tokens, which are the building blocks of the text data
- Let's apply this concept to the Kite example

What’s the denominators?

- We have a text corpus about “Kite” from Wikipedia
  - 2 documents: Intro and History
- How did we calculate Term Frequency of "kite" in each document?
- What if we divide each "kite” by the number of tokens in the corpus?
  - If so, how do we interpret the number?
- Let’s go back to the code

Term’s inverse document frequency

- We calculate the ratio of the total number of documents to the number of documents the term appears in
  - Called a term’s inverse document frequency
- What’s the motivation?
  - Term frequency alone (normalized frequency of a word) is not enough

Example

- There are two documents
  - Both documents contain the words “and” and “kite”
  - Only one document contains the word “china”
- So, IDF of
  - ”and”: 1, “kite”: 1, ”china”: 2
  - Make sure of what it means!

Think about this statement in the textbook

A good way to think of a term’s inverse document frequency is this:

How strange is it that this token is in this document?

If a term appears in one document a lot of times, but occurs rarely in the rest of the corpus, one could assume it’s important to that document specifically.

IDF itself is not document-specific number

- IDF of three tokens in the Kite Corpus
  - ”and”: 1, “kite”: 1, ”china”: 2
  - It does not change by documents
  - It does not mean that the word “china” is rarely used
  - Even if one document contains the word “china” one million times and the other document does not contain it, IDFs are the same

TF-IDF

- Previously, we only focus on the frequency for each word (term)
- Now, for a given term t, in a given document d in a corpus D
  - TF-IDF(t,d,D) = tf(t,d) * idf(t,D)
  - The base of log function is not important
  - Assign weights to words (terms) based on
    - Frequency of words in a certain document
    - Distribution of words over documents in the corpus
- You won’t likely ever have to implement the preceding formulas for computing TF-IDF

Interpretation of TF-IDF

- The more times a word appears in the document, the TF (and hence the TF-IDF) will go up
- At the same time, as the number of documents that contain that word goes up, the IDF (and hence the TF-IDF) for that word will go down
- It relates a specific word or token to a specific document in a specific corpus, and then it assigns a numeric value to the importance of that word **in the given document, given its usage across the entire corpus**.

Is this topic modeling?

- TF-IDF is used to identify the most important words or terms in a document or corpus by assigning weights to them
- It presents the word’s rarity across the corpus
- Possible to collect or filter the keywords based on the distribution of TF-IDF
    - Are they (important words) topics?

Relevance ranking

- Let’s go back to the code again
- Let’s explain in words what this example describes
- What’s 0 mean?
- What’s 0.5235048549676834 mean?
- Let’s revisit the Question 5 on Homework 1