Dan Jurafsky and James Martin

Speech and Language Processing

Chapter 6:
Vector Semantics

Representing strings with vectors of words or characters

- *to be or not to be a bee is the question*, said the queen bee

Multiset
a
be
be
bee
bee
is
not
or
queen
question
said
the
the
to
to

Word frequencies	
Words	Count
a	1
be	2
bee	2
is	1
not	1
or	1
queen	1
question	1
said	1
the	2
to	2

Word frequencies		
Words	Raw Count	Relative Frequency
a	1	0.07
be	2	0.13
bee	2	0.13
is	1	0.07
not	1	0.07
or	1	0.07
queen	1	0.07
question,	1	0.07
said	1	0.07
the	2	0.13
to	2	0.13
Total	15	1

More context?

- *to be or not to be a bee is the question*, said the queen bee

Multiset
to be
be or
or not
not to
to be
be a
a bee
bee is
is the
the question
question said
said the
the queen
queen bee

Word pair frequencies	
Word pairs	Count
a bee	1
be a	1
be or	1
bee is	1
is the	1
not to	1
or not	1
queen bee	1
question said	1
said the	1
the queen	1
the question	1
to be	2

Word pair frequencies		
Word pairs	Raw Count	Relative Frequency
a bee	1	0.07
be a	1	0.07
be or	1	0.07
bee is	1	0.07
is the	1	0.07
not to	1	0.07
or not	1	0.07
queen bee	1	0.07
question said	1	0.07
said the	1	0.07
the queen	1	0.07
the question	1	0.07
to be	2	0.14
Total	14	1

More context?

- *to be or not to be a bee is the question*, said the queen bee
- Unigrams
- Bigrams
- Trigrams
- 4 grams and so on…

Character-level vectors

- *to be or not to be a bee is the question*, said the queen bee

Character frequencies		
Characters	Raw Count	Relative Frequency
a	2	0.04
b	4	0.09
d	1	0.02
e	11	0.24
h	2	0.04
i	3	0.07
n	3	0.07
o	5	0.11
q	2	0.04
r	1	0.02
s	3	0.07
t	6	0.13
u	2	0.04
Total	45	1

Character pair frequencies		
Character pairs	Raw Count	Relative Frequency
ab	1	0.02
ai	1	0.02
be	4	0.09
dt	1	0.02
ea	1	0.02
ee	3	0.07
ei	1	0.02
en	1	0.02
eo	1	0.02
eq	2	0.05
es	1	0.02
he	2	0.05
id	1	0.02
io	1	0.02
is	1	0.02
nb	1	0.02
no	1	0.02
ns	1	0.02
ob	2	0.05
on	1	0.02
or	1	0.02
ot	1	0.02
qu	2	0.05
rn	1	0.02
sa	1	0.02
st	2	0.05
th	2	0.05
ti	1	0.02
to	2	0.05
tt	1	0.02
ue	2	0.05
Total	44	1.00

Bag of Words

- A simplifying representation
- Text (such as a sentence or a document) is represented as the bag (multiset) of its words
  - Disregarding grammar and even word order but keeping multiplicity
- Used in document classification where the (frequency of) occurrence of each word is used as a feature for training a classifier
- But some words are common in general which do not tell a lot about a document (e.g. the, is, a)
- TF-IDF

Measures to normalize term-frequencies

- **Raw frequency**: The number of times that term *t* occurs in document *d*,
  - ***tf(t,d) = $f_{t,d}$***
  - We need to remove bias towards long or short documents
    - Normalize by document length
- **Relative term frequency i.e. tf adjusted for document length**:
  - *tf(t,d)* = $f_{t,d}$ ÷ (number of words in d)

But raw frequency is a bad representation

Frequency is clearly useful; if *sugar* appears a lot near *apricot*, that's useful information.

But overly frequent words like *the*, *it,* or *they* are not very informative about the context

Need a function that resolves this frequency paradox!

Other measures to normalize term-frequencies

- Next, we need ways to remove bias towards more frequently occurring words.
  - A word appearing 100 times in a document does not make it a 100 times more likely representative of the document

- **Boolean** ”frequency": tf(*t*,*d*) = 1 if *t* occurs in *d* and 0 otherwise

- **Logarithmically scaled term frequency**:

$$\mathrm{tf}_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{count}(t,d) & \text{if } \mathrm{count}(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

  - Thus terms which occur 10 times in a document would have a *tf*=2, 100 times in a document *tf*=3, 1000 times *tf*=4,...

TF-IDF

- TF-IDF increases proportionally to the number of times a word appears in the *document*,
- It is offset by the frequency of the word in the whole *corpus of documents*
- Helps adjust for the fact that some words appear more frequently in general.

IDF

- Give a higher weight to words that occur only in a few documents
- Terms that are limited to a few documents are useful for discriminating those documents from the rest of the collection; terms that occur frequently across the entire collection aren't as helpful.
- Because of the large number of documents in many collections, this measure is usually squashed with a log function.
- **Inverse document frequency, idf(t, d)** is the logarithmically scaled inverse fraction of the documents that contain the word

$$\operatorname{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

- N: total number of documents in the corpus $N = |D|$
- $|\{d \in D : t \in d\}|$: number of documents where the term t appears (i.e., $\operatorname{tf}(t, d) \neq 0$). If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to $1 + |\{d \in D : t \in d\}|$.

For example, if we have a corpus of 100 documents, and the word "apple" appears in 20 of those documents, the IDF for "apple" would be: idf = log(100 / 20) = log(5)

tf-idf: combine two factors

tf: term frequency. frequency count (usually log-transformed):

$$\mathrm{tf}_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{count}(t,d) & \text{if } \mathrm{count}(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Idf: inverse document frequency: tf-

$$\mathrm{idf}_i = \log\left(\frac{N}{\mathrm{df}_i}\right)$$

Total # of docs in collection

# of docs that have word i

Words like "the" or "good" have very low idf

tf-idf value for word t in document d:

$$w_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t$$

TF-IDF

Here are some idf values for some words in the Shakespeare corpus, ranging from extremely informative words which occur in only one play like *Romeo*, to those that occur in a few like *salad* or *Falstaff*, to those which are very common like *fool* or so common as to be completely non-discriminative since they occur in all 37 plays like *good* or *sweet*.[3]

Word	**df**	**idf**
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.074
fool	36	0.012
good	37	0
sweet	37	0

TF-IDF

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.074
fool	36	0.012
good	37	0
sweet	37	0

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

Figure 6.8 A tf-idf weighted term-document matrix for four words in four Shakespeare plays, using the counts in Fig. 6.2. Note that the idf weighting has eliminated the importance of the ubiquitous word *good* and vastly reduced the impact of the almost-ubiquitous word *fool*.

TF-IDF

https://en.wikipedia.org/wiki/Tf%E2%80%93idf

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

$$\mathrm{tf}(\text{"this"}, d_1) = \frac{1}{5} = 0.2$$

$$\mathrm{tf}(\text{"this"}, d_2) = \frac{1}{7} \approx 0.14$$

$$\mathrm{idf}(\text{"this"}, D) = \log\left(\frac{2}{2}\right) = 0$$

$$\mathrm{tfidf}(\text{"this"}, d_1, D) = 0.2 \times 0 = 0$$

$$\mathrm{tfidf}(\text{"this"}, d_2, D) = 0.14 \times 0 = 0$$

$$\mathrm{tf}(\text{"example"}, d_1) = \frac{0}{5} = 0$$

$$\mathrm{tf}(\text{"example"}, d_2) = \frac{3}{7} \approx 0.429$$

$$\mathrm{idf}(\text{"example"}, D) = \log\left(\frac{2}{1}\right) = 0.301$$

$$\mathrm{tfidf}(\text{"example"}, d_1, D) = \mathrm{tf}(\text{"example"}, d_1) \times \mathrm{idf}(\text{"example"}, D) = 0 \times 0.301 = 0$$

$$\mathrm{tfidf}(\text{"example"}, d_2, D) = \mathrm{tf}(\text{"example"}, d_2) \times \mathrm{idf}(\text{"example"}, D) = 0.429 \times 0.301 \approx 0.13$$

Now that we have our vectors!

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Figure 6.3 The term-document matrix for four words in four Shakespeare plays. The red boxes show that each document is represented as a column vector of length four.

– Calculate the distance between vectors

Vectors are the basis of information retrieval

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Vectors are similar for the two comedies
Different than the history

Comedies have more fools and wit and fewer battles.

Words can be vectors too

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

battle is "the kind of word that occurs in Julius Caesar and Henry V"

fool is "the kind of word that occurs in comedies, especially Twelfth Night"

Angles between vectors

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Figure 6.3 The term-document matrix for four words in four Shakespeare plays. The red boxes show that each document is represented as a column vector of length four.

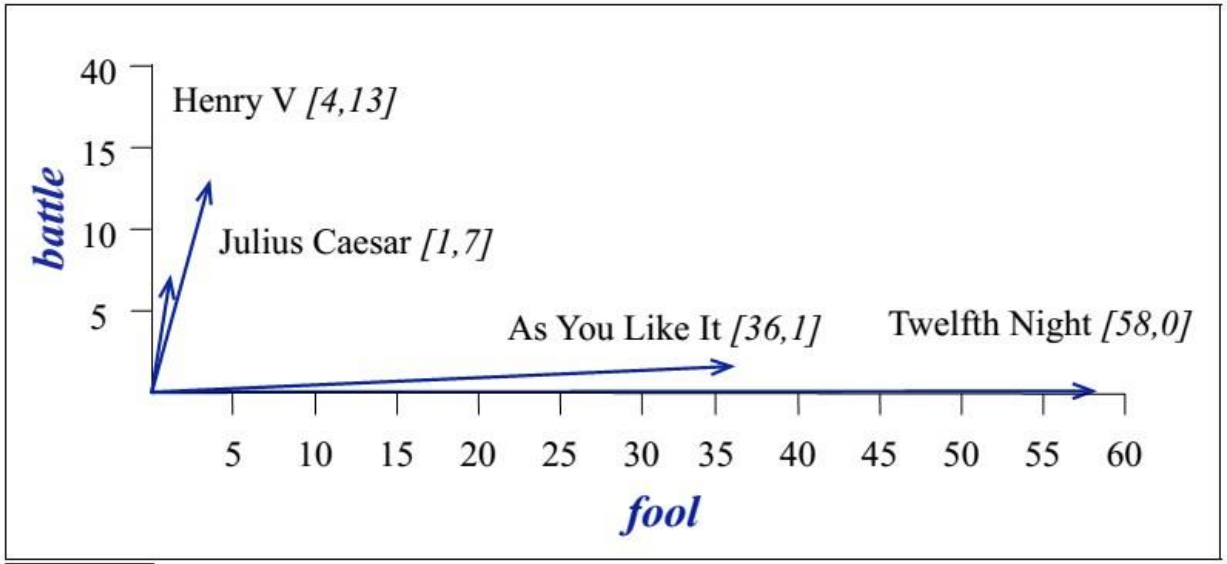

Figure 6.4 A spatial visualization of the document vectors for the four Shakespeare play documents, showing just two of the dimensions, corresponding to the words *battle* and *fool*. The comedies have high values for the *fool* dimension and low values for the *battle* dimension.

Cosine Similarity

$$\text{dot-product}(\vec{v},\vec{w}) = \vec{v}\cdot\vec{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_N w_N$$

- This raw dot-product, however, has a problem as a similarity metric: it favors **vector length long** vectors.
- The simplest way to modify the dot product to normalize for the vector length is to divide the dot product by the lengths of each of the two vectors.
- This **normalized dot product** turns out to be the same as the cosine of the angle between the two

$$\vec{a}\cdot\vec{b} = |\vec{a}||\vec{b}|\cos\theta$$

$$\frac{\vec{a}\cdot\vec{b}}{|\vec{a}||\vec{b}|} = \cos\theta \qquad (6.9)$$

The **cosine** similarity metric between two vectors $\vec{v}$ and $\vec{w}$ thus can be computed as:

$$\text{cosine}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}} \qquad (6.10)$$

Cosine Similarity

- The cosine value ranges from 1 for vectors pointing in the same direction, through 0 for vectors that are orthogonal, to -1 for vectors pointing in opposite directions.
- But raw frequency values are non-negative, so the cosine for these vectors ranges from 0–1.

Cosine Similarity

	large	data	computer
apricot	2	0	0
digital	0	1	2
information	1	6	1

$$\cos(\text{apricot},\text{information}) = \frac{2+0+0}{\sqrt{4+0+0}\sqrt{1+36+1}} = \frac{2}{2\sqrt{38}} = .16$$

$$\cos(\text{digital},\text{information}) = \frac{0+6+2}{\sqrt{0+1+4}\sqrt{1+36+1}} = \frac{8}{\sqrt{38}\sqrt{5}} = .58 \quad (6.11)$$

The model decides that *information* is closer to *digital* than it is to *apricot*, a result that seems sensible. Fig. 6.7 shows a visualization.

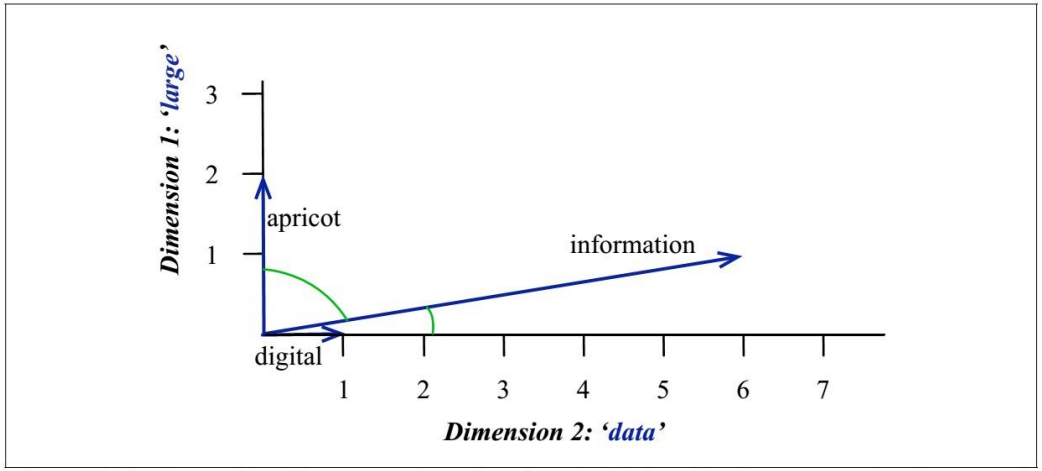

Figure 6.7 A graphical demonstration of cosine similarity, showing vectors for three words (*apricot*, *digital*, and *information*) in the two dimensional space defined by counts of the words *data* and *large* in the neighborhood. Note that the angle between *digital* and *information* is smaller than the angle between *apricot* and *information*. When two vectors are more similar, the cosine is larger but the angle is smaller; the cosine has its maximum (1) when the angle between two vectors is smallest (0°); the cosine of all other angles is less than 1.

All Distances

S1: *to be or not to be a bee is the question*, said the queen bee

S2: *one needs to be strong in order to be a queen bee*

Words	# S1	# S2	f1	f2	sqrt((f1 - f2)^2)	|f1 - f2|	Max |f1-f2|	f1 . f2	|f1|	|f2|	f1.f2/|f1||f2|
a	1	1	0.0667	0.0833	0.0003	0.0167	0.0167	0.0056	0.0044	0.0069	
be	2	2	0.1333	0.1667	0.0011	0.0333	0.0333	0.0222	0.0178	0.0278	
bee	2	1	0.1333	0.0833	0.0025	0.0500	0.0500	0.0111	0.0178	0.0069	
in	0	1	0.0000	0.0833	0.0069	0.0833	0.0833	0.0000	0.0000	0.0069	
is	1	0	0.0667	0.0000	0.0044	0.0667	0.0667	0.0000	0.0044	0.0000	
needs	0	1	0.0000	0.0833	0.0069	0.0833	0.0833	0.0000	0.0000	0.0069	
not	1	0	0.0667	0.0000	0.0044	0.0667	0.0667	0.0000	0.0044	0.0000	
one	0	1	0.0000	0.0833	0.0069	0.0833	0.0833	0.0000	0.0000	0.0069	
or	1	0	0.0667	0.0000	0.0044	0.0667	0.0667	0.0000	0.0044	0.0000	
order	0	1	0.0000	0.0833	0.0069	0.0833	0.0833	0.0000	0.0000	0.0069	
queen	1	1	0.0667	0.0833	0.0003	0.0167	0.0167	0.0056	0.0044	0.0069	
question	1	0	0.0667	0.0000	0.0044	0.0667	0.0667	0.0000	0.0044	0.0000	
said	1	0	0.0667	0.0000	0.0044	0.0667	0.0667	0.0000	0.0044	0.0000	
strong	0	1	0.0000	0.0833	0.0069	0.0833	0.0833	0.0000	0.0000	0.0069	
the	2	0	0.1333	0.0000	0.0178	0.1333	0.1333	0.0000	0.0178	0.0000	
to	2	2	0.1333	0.1667	0.0011	0.0333	0.0333	0.0222	0.0178	0.0278	
Total	15	12	1	1	0.2828	1.0333	0.1333	0.0667	0.3197	0.3333	
					Euclidean	Manhattan	Chebyshev	Dot product			Cosine Similarity

Automatic Thesaurus Generation

Attempt to generate a thesaurus automatically by analyzing a collection of documents

Fundamental notion: similarity between two words

Definition 1: Two words are similar if they co-occur with similar words.

Definition 2: Two words are similar if they occur in a given grammatical relation with the same words.

You can harvest, peel, eat, prepare, etc. apples and pears, so apples and pears must be similar.

Co-occurrence based is more robust, grammatical relations are more accurate.

Words, Lemmas, Senses, Definitions

lemma **sense** **definition**

pepper, *n.*

Pronunciation: Brit. /pɛpə/, U.S. /ˈpɛpər/

Forms: OE **peopor** (*rare*), OE **pipcer** (transmission error), OE **pipor**, OE **pipur** (*rare*

Frequency (in current use):

Etymology: A borrowing from Latin. **Etymon:** Latin *piper*.
< classical Latin *piper*, a loanword < Indo-Aryan (as is ancient Greek *πέπερι*); compare Sa

I. The spice or the plant.

1.

a. A hot pungent spice derived from the prepared fruits (peppercorns) of the pepper plant, *Piper nigrum* (see sense 2a), used from early times to season food, either whole or ground to powder (often in association with salt). Also (locally, chiefly with distinguishing word): a similar spice derived from the fruits of certain other species of the genus *Piper*; the fruits themselves.

The ground spice from *Piper nigrum* comes in two forms, the more pungent *black pepper*, produced from black peppercorns, and the milder *white pepper*, produced from white peppercorns: see BLACK *adj.* and *n.* Special uses 5a, PEPPERCORN *n.* 1a, and WHITE *adj.* and *n.*[1] Special uses 7b(a).

2.

a. The plant *Piper nigrum* (family Piperaceae), a climbing shrub indigenous to South Asia and also cultivated elsewhere in the tropics, which has alternate stalked entire leaves, with pendulous spikes of small green flowers opposite the leaves, succeeded by small berries turning red when ripe. Also more widely: any plant of the genus *Piper* or the family Piperaceae.

b. Usu. with distinguishing word: any of numerous plants of other families having hot pungent fruits or leaves which resemble pepper (1a) in taste and in some cases are used as a substitute for it.

c. *U.S.* The California pepper tree, *Schinus molle*. Cf. PEPPER TREE *n.*

3. Any of various forms of capsicum, esp. *Capsicum annuum* var. *annuum*. Originally (chiefly with distinguishing word): any variety of the *C. annuum* Longum group, with elongated fruits having a hot, pungent taste, the source of cayenne, chilli powder, paprika, etc., or of the perennial *C. frutescens*, the source of Tabasco sauce. Now frequently (more fully ***sweet pepper***): any variety of the *C. annuum* Grossum group, with large, bell-shaped or apple-shaped, mild-flavoured fruits, usually ripening to red, orange, or yellow and eaten raw in salads or cooked as a vegetable. Also: the fruit of any of these capsicums.

Sweet peppers are often used in their green immature state (more fully ***green pepper***), but some new varieties remain green when ripe.

Lemma pepper

Sense 1: spice from pepper plant

Sense 2: the pepper plant itself

Sense 3: another similar plant (Jamaican pepper)

Sense 4: another plant with peppercorns (California pepper)

Sense 5: *capsicum* (i.e. chili, paprika, bell pepper, etc)

A sense or “concept” is the meaning component of a word

Let's define words by their usages

In particular, words are defined by their environments (the words around them)

Zellig Harris (1954): **If A and B have almost identical environments we say that they are synonyms**.

What does ong choi mean?

Suppose you see these sentences:

- Ongchoi is delicious **sautéed with garlic**.
- Ongchoi is superb **over rice**
- Ongchoi **leaves** with salty sauces

And you've also seen these:

- …spinach **sautéed with garlic over rice**
- Chard stems and **leaves** are **delicious**
- Collard greens and other **salty** leafy greens

Conclusion:

- Ongchoi is a leafy green like spinach, chard, or collard greens

Ong choi: *Ipomoea aquatica "Water Spinach"*

Yamaguchi, Wikimedia Commons, public domain

Example of Tf*Idf Vector

Represent the word “apple” as vector using following corpus. Use TF.IDF weights. Assume the window size for word context is 2

Document 1: I like to ride cycle often.

Document 2: Ali and Hassan ate apple and oranges.

Document 3: Ali ate apple not oranges

Example of Tf*Idf Vector

Represent the word “apple” as vector using following corpus. Use TF.IDF weights. Assume the window size for word context is 2

Document 1: I like to ride cycle often.

Document 2: Ali and Hassan ate apple and oranges.

Document 3: Ali ate apple not oranges

Context words: Hassan, ate, and, oranges,

Ali, not

Step 1: Term Frequency (TF)

First, calculate the term frequency (TF) for each context word in each document.

TF is calculated as:

$$\text{TF}(\text{word}, \text{doc}) = \frac{\text{Number of occurrences of the word in the document}}{\text{Total number of words in the document}}$$

Document 2: "Ali and [Hassan ate apple and oranges]"

- Total words in the document: 7
  - TF(Hassan) = $\frac{1}{7}$
  - TF(ate) = $\frac{1}{7}$
  - TF(and) = $\frac{1}{7}$
  - TF(oranges) = $\frac{1}{7}$

Document 3: "[Ali ate apple not oranges]"

- Total words in the document: 6
  - TF(Ali) = $\frac{1}{6}$
  - TF(ate) = $\frac{1}{6}$
  - TF(not) = $\frac{1}{6}$
  - TF(oranges) = $\frac{1}{6}$

Step 2: Inverse Document Frequency (IDF)

Now, compute the inverse document frequency (IDF) for each context word.

IDF is calculated as:

$$\text{IDF}(\text{word}) = \log\left(\frac{\text{Total number of documents}}{\text{Number of documents containing the word}}\right)$$

There are 3 documents in total.

IDF for each word:

- **Hassan** appears in Document 2.
  - IDF(Hassan) = $\log\left(\frac{3}{1}\right) = \log(3) \approx 1.098$
- **ate** appears in Document 2 and Document 3.
  - IDF(ate) = $\log\left(\frac{3}{2}\right) = \log(1.5) \approx 0.405$
- **and** appears in Document 2.
  - IDF(and) = $\log\left(\frac{3}{1}\right) = \log(3) \approx 1.098$
- **oranges** appears in Document 2 and Document 3.
  - IDF(oranges) = $\log\left(\frac{3}{2}\right) = \log(1.5) \approx 0.405$
- **Ali** appears in Document 3.
  - IDF(Ali) = $\log\left(\frac{3}{1}\right) = \log(3) \approx 1.098$
- **not** appears in Document 3.
  - IDF(not) = $\log\left(\frac{3}{1}\right) = \log(3) \approx 1.098$

Step 3: Calculate TF-IDF

Now multiply the TF by the corresponding IDF for each context word.

Document 2:

- TF-IDF(Hassan) = $\frac{1}{7} \times 1.098 = 0.157$
- TF-IDF(ate) = $\frac{1}{7} \times 0.405 = 0.058$
- TF-IDF(and) = $\frac{1}{7} \times 1.098 = 0.157$
- TF-IDF(oranges) = $\frac{1}{7} \times 0.405 = 0.058$

Document 3:

- TF-IDF(Ali) = $\frac{1}{6} \times 1.098 = 0.183$
- TF-IDF(ate) = $\frac{1}{6} \times 0.405 = 0.068$
- TF-IDF(not) = $\frac{1}{6} \times 1.098 = 0.183$
- TF-IDF(oranges) = $\frac{1}{6} \times 0.405 = 0.068$

Step 4: Construct the TF-IDF vector for "apple“
combine all the TF-IDF values for the context words to create the vector for "apple“
TF-IDF vector=[0.157, **0.063**, 0.157, 0.058, 0.183, 0.068, 0.183]
*Hassan **ate** and oranges ali **ate** not oranges*

Notice ‘ate’ comes twice: so in the consolidated vector u can

- Average tf-idf value of ‘ate’
- Weighted average tf-idf value of ate

Summary: tf-idf

Compare two words using tf-idf cosine to see if they are similar

Compare two documents

- Take the centroid of vectors of all the words in the document
- Centroid document vector is:

$$d = \frac{w_1 + w_2 + \ldots + w_k}{k}$$

Where:

- $\vec{w}_1, \vec{w}_2, \ldots, \vec{w}_k$ are the TF-IDF vectors of all the words in the document.
- k is the number of words in the document.

1. **Counting repeated words** (Approach 1) gives more weight to frequent words, emphasizing their importance in the document.2.
2. **Counting unique words** (Approach 2) reduces the impact of word frequency, focusing on the diversity of words.

tf-idf cosine

When comparing two documents, we treat the document as a collection of word vectors, and to get a single vector for a document, we calculate its **centroid**.

The centroid is simply the average of the TF-IDF vectors of all the words in the document.

By comparing centroids, you can measure the overall similarity of two documents rather than comparing individual words.

- **Word Comparison**: Use cosine similarity between TF-IDF vectors of words.
- **Document Comparison**: Compute the centroid of the TF-IDF vectors of all words in the document, and then compare centroids using cosine similarity.

For document cosine similarity: f1.f2/|f1||f2|