Natural Language Processing

TF-IDF and Word Vectors

Ramaseshan Ramachandran

WORDS AND TERMS

The atomic unit is a ***word***

- Words are built from the alphabet of a language; we treat the ***word*** (or ***term***, which may span co-occurring words) as the atomic unit
- For computation, every word needs a numerical representation
- The vocabulary $V = \{w_1, w_2, w_3, \ldots, w_n\}$ is the set of the n unique words of a corpus
- A word of V may appear in a document of the collection $D = \{D_1, D_2, D_3, \ldots, D_m\}$ once, several times, or not at all

TERM-FREQUENCY WEIGHTING SCHEMES

$$\text{Boolean} - 0, 1 \tag{1}$$

$$\text{Raw count} - \mathrm{tf}_{i,d} \tag{2}$$

$$\text{Adjusted to document length} - \frac{\mathrm{tf}_{i,d}}{M_d} \tag{3}$$

$$\text{Log weighting} - \begin{cases} 1 + \log \mathrm{tf}_{i,d} & \text{if } \mathrm{tf}_{i,d} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where $\mathrm{tf}_{i,d}$ is the count of term i in document d and M_d is the number of tokens in d

DISADVANTAGES OF RAW FREQUENCY

- All terms are treated as equally important
- A common term such as ***the*** carries no information about the document, yet receives a high score
- Classification suffers when the score is dominated by common words shared across documents

INVERSE DOCUMENT FREQUENCY

We want to scale down the weight of frequently occurring terms and, at the same time, boost the weight of rarely occurring terms.
Inverse document frequency (IDF) is defined as

$$\mathrm{IDF}_t = \log\left(\frac{N}{df_t}\right) \tag{5}$$

where N is the total number of documents in the collection, and df_t is the number of documents containing the term t

- Rare terms get a significantly higher weight
- Commonly occurring terms are attenuated
- It is a measure of informativeness
- The tf weight of a term is reduced by a factor that grows with the number of documents it appears in
- If a term appears in every document, its IDF is zero. The term does not discriminate between documents

TF-IDF

Composition of TF and IDF produces a composite scaling for each term in the document

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_{t,d} \tag{6}$$

- The value is high when t occurs many times within a few documents
- The value is very low when a term appears in all documents

INVERSE DOCUMENT FREQUENCY-IDF

IDF measures how rare the term t is across the document collection. It is computed as

$$\text{IDF of a term t} = \log_{10}\left(\frac{\text{Total number of documents in a corpus}}{\text{Count of documents with term t}}\right)$$

IDF is the measure of ***informativeness***

Example:

Consider a corpus with 100K documents. The word ***moon*** occurs in some documents (say, 100) with the following frequency:

$\text{TF}_{d_1} = \frac{20}{427}, \text{TF}_{d_2} = \frac{30}{250}, \text{TF}_{d_3} = \frac{20}{250}, \text{TF}_{d_9} = \frac{5}{125}$ and $\text{TF}_{d_{1000}} = \frac{20}{1000}$

The total number of documents in the corpus $= 100{,}000$

$$\therefore \text{IDF}_{d_1} = \log_{10}\left(\frac{100000}{100}\right)$$

$$\text{TF}_{d_1} * \text{IDF} = 0.141$$

If the word ***Andromeda*** appears only once d_1, then $\text{TF}_{d_1} * \text{IDF} = 0.0117$. If the word ***the*** appeared in every document and 45 times in d1, then $\text{TF} * \text{IDF} = 0.210$

DOCUMENT RANKING USING TF-IDF

Using TF-IDF, the documents can be rank-ordered for the query term *moon*.

Document Name	tf-idf	Rank
d1	0.14	3
d2	0.36	1
d3	0.24	2
d9	0.12	4
d1000	0.06	5

BAG OF WORDS

Representing a document by its term weights alone, $[\mathrm{tf}_1, \mathrm{tf}_2, \mathrm{tf}_3, \ldots, \mathrm{tf}_n]$, is known as the *bag-of-words* model

- The order of the terms is ignored. Only their counts matter
- Two documents with similar bags of words are assumed to be similar in content
- It is a purely quantitative representation of the document
- Contrast: modern contextual models (transformers) explicitly model word order through position information

OPEN-CLASS AND STOP WORDS

Open-class words: Carry the main semantic content of a sentence

- Contribute to the overall meaning of a sentence
- Include nouns, verbs, adjectives, and adverbs
- **Expansive**: New words can be added to this category over time (e.g., "vlog," "emoji", “Mulligatawny”)
- **Adaptability**: New terms to express emerging concepts, technologies, and innovation

Closed-class (function) words, the usual stop words, serve grammatical purposes

- Include
  - prepositions (in, on, with, ...)
  - conjunctions (and, but, or..)
  - articles (a, an, the)
  - pronouns(he, she, they...)
- Essential for the grammatical integrity of sentences
- Contribute less to the overall meaning of the sentence

WHAT IS LEXICAL SIMILARITY?

- Measures similarity based on exact words or surface forms.
- Focuses on word overlap or structural resemblance.
- Does not consider meaning or context.

Lexically similar sentences

1. The dog bites the man
2. The man bites the dog

In the bag of words model, these two sentences are exactly the same - perfect match - disregards grammar and order of words

Techniques to Measure Lexical Similarity

JACCARD SIMILARITY

- Jaccard Similarity measures the similarity between two sets.
- Defined as the size of the intersection divided by size of the union:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

- Values range from 0 (no similarity) to 1 (identical sets).

Example: Two sentences:

- "The dog bites the man"
- "The man bites the dog"

$$J(A,B) = \frac{4}{4} = 1$$

These sentences have identical word sets despite differing word order.

COSINE SIMILARITY

- Cosine Similarity measures the cosine of the angle between two vectors.
- For text, vectors represent word frequency counts.
- Formula:

$$\text{Cosine Similarity}(A, B) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_i A_i B_i}{\sqrt{\sum_i A_i^2}\sqrt{\sum_i B_i^2}}$$

$$\text{Cosine Distance} = 1 - \text{Cosine Similarity}$$

- Cosine similarity Value ranges from 0 (orthogonal) to 1 (identical vectors).
- Cosine Distance Value ranges from 0 (identical) to 1 (orthogonal vectors).

COSINE SIMILARITY: EXAMPLE

Example sentences:

- "The dog bites the man"
- "The man bites the dog."

Words considered (lowercase): the, dog, bites, man
Frequency vectors:

$$A = [2,1,1,1] \quad (\text{the} = 2, \text{dog} = 1, \text{bites} = 1, \text{man} = 1)$$

$$B = [2,1,1,1] \quad (\text{the} = 2, \text{dog} = 1, \text{bites} = 1, \text{man} = 1)$$

$$\text{Cosine Similarity} = \frac{7}{\sqrt{7} \times \sqrt{7}} = \frac{7}{7} = 1$$

EDIT DISTANCE SIMILARITY

- Edit Distance (Levenshtein Distance) measures the minimum number of single-character edits (insertions, deletions, or substitutions) required to change one string into another.
- Edit Distance Similarity converts this distance into a similarity score.
- Formula (normalized similarity):

$$\text{Similarity} = 1 - \frac{\text{Edit Distance}}{\max(\text{length}(A), \text{length}(B))}$$

- Values range from 0 (completely different) to 1 (identical strings).

EDIT DISTANCE SIMILARITY: EXAMPLE (IGNORING PUNCTUATION)

Example sentences:

- "The dog bites the man"
- "The man bites the dog"

As lowercase strings without punctuation:

$$\mathrm{A} = \texttt{thedogbitestheman}, \quad \mathrm{B} = \texttt{themanbitesthedog}$$

Length of both strings: 17 characters.
Edit operations (substitutions): 6

$$\text{similarity} = 1 - \frac{6}{17} \approx 0.65$$

COMPUTING EDIT DISTANCE

Formula to Compute Edit Distance is

$$D(i+1,j+1) = \min \begin{cases} D(i,j+1)+1 & \text{(Deletion)} \\ D(i+1,j)+1 & \text{(Insertion)} \\ D(i,j)+\text{cost} & \text{(Substitution)} \end{cases} \qquad \text{cost} = \begin{cases} 0 & \text{if } S[i+1] = T[j+1] \\ 1 & \text{if } S[i+1] \neq T[j+1] \end{cases} \tag{7}$$

Table: Dynamic Programming Matrix for Edit Distance between “Statistics” and “Probability”

		P	**r**	**o**	**b**	**a**	**b**	**i**	**l**	**i**	**t**	**y**
	0	1	2	3	4	5	6	7	8	9	10	11
S	1	1	2	3	4	5	6	7	8	9	10	11
t	2	2	2	3	4	5	6	7	8	9	9	10
a	3	3	3	3	4	4	5	6	7	8	9	10
t	4	4	4	4	4	5	5	6	7	8	8	9
i	5	5	5	5	5	5	6	5	6	7	8	9
s	6	6	6	6	6	6	6	6	6	7	8	9
t	7	7	7	7	7	7	7	7	7	7	7	8
i	8	8	8	8	8	8	7	7	8	7	8	8
c	9	9	9	9	9	9	8	8	8	8	8	9
s	10	10	10	10	10	10	9	9	9	9	9	**9**

APPLICATIONS OF LEXICAL SIMILARITY

- Plagiarism detection
- Document clustering based on word overlap
- Keyword-based search engines
- Data deduplication

WHAT ARE SEMANTIC RELATIONSHIPS?

Semantic relationships are connections between words or phrases based on meaning, structure, or context

- Enable deeper language understanding.
- Useful for search, translation, and sentiment analysis

TYPES OF SEMANTIC RELATIONSHIPS

- Synonymy
- Antonymy
- Hyponymy/Hypernymy
- Meronymy/Holonymy
- Polysemy/Homonymy
- Troponymy
- Converseness
- Cause-Effect

SYNONYMY AND ANTONYMY

Synonymy: Words with similar meanings.

- Example: happy $\leftrightarrow$ joyful

Antonymy: Words with opposite meanings.

- Example: hot $\leftrightarrow$ cold

HYPONYMY/HYPERNYMY

- **Hyponym**: More specific term. Example: sandwich, fruit
- **Hypernym**: More general term. Example: food (includes sandwich, fruit)

MERONYMY/HOLONYMY

- **Meronymy**: Part-whole relationship. Example: wheel (part of) car
- **Holonymy**: Whole-part relationship. Example: car (has) wheel

POLYSEMY/HOMONYMY

- **Polysemy**: Word with multiple related meanings. Example: bank (financial), bank (river side) , bank (have faith), bank (flight maneuver)
- **Homonymy**: Word with unrelated meanings but same spelling or sound. Example: light (not heavy), light (illumination)

OTHER RELATIONSHIPS

- **Troponymy**: Specifies manner or degree (run → jog)
  Drizzle vs cats and dogs (wrt to rain)
- **Converseness**: Reciprocal relationships (buy/sell; parent/child)
- **Cause-Effect**: Indicates causality (heat causes melting)

APPLICATIONS

- Sentence similarity
- Paraphrase generation
- Word sense disambiguation
- Knowledge graphs and ontologies

LEXICAL VS SEMANTIC SIMILARITY

- Lexical similarity measures text resemblance based on surface-level features like exact word matches or string overlap, without considering meaning.
- Semantic similarity evaluates deeper contextual and conceptual closeness using techniques like word embeddings to account for synonyms, intent, and relationships.

KEY DIFFERENCES

"cat sat on mat" "feline rested on rug"
"cat chases the dog" "dog chases the cat"

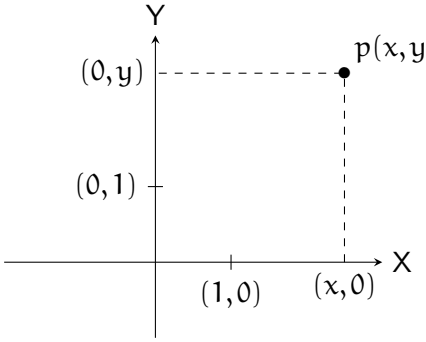
Feature	Lexical Similarity	Semantic Similarity
Basis	Exact word match	Meaning and context
Techniques	Jaccard, edit distance	Embeddings, cosine similarity
Strength	Fast, precise	Flexible, meaning-aware
Weakness	Surface only	Computational cost

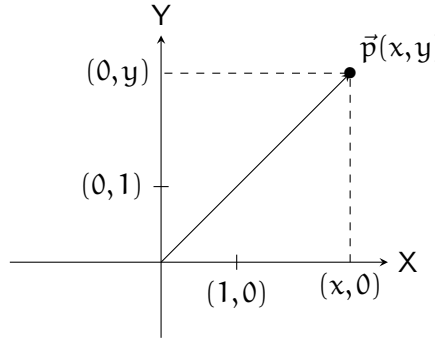
USE CASES

- Lexical: Keyword search, plagiarism detection.
- Semantic: NLP systems, recommendations, retrieval-augmented generation.

2-D VECTOR SPACE

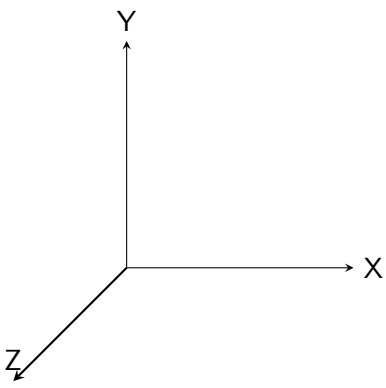
A 2-D vector-space is defined as a set of linearly independent basis vectors with 2 axes. Each axis corresponds to a dimension in the vector-space

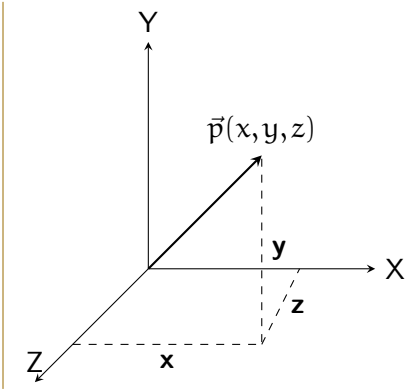




3-D VECTOR SPACE

A 3-D vector-space is defined as a set of linearly independent basis vectors with 3 axes. Each axis corresponds to a dimension in the vector-space





Linearly independent vectors of size $\mathcal{N}$ will result in $\mathcal{N}$-dimensional axes which are mutually orthogonal to each other

VECTOR SPACE MODEL FOR WORDS

Let us assume that the words in a corpus are considered as linearly independent basis vectors.

VECTOR SPACE MODEL FOR WORDS

Let us assume that the words in a corpus are considered as linearly independent basis vectors.
If a corpus contains $|\mathbb{V}|$ words which are linearly independent, then every word represents an axis in the continuous vector space $\mathbb{R}$.

VECTOR SPACE MODEL FOR WORDS

Let us assume that the words in a corpus are considered as linearly independent basis vectors.
If a corpus contains $|\mathbb{V}|$ words which are linearly independent, then every word represents an axis in the continuous vector space $\mathbb{R}$.
Each word takes an independent axis which is orthogonal to other words/axes.

VECTOR SPACE MODEL FOR WORDS

Let us assume that the words in a corpus are considered as linearly independent basis vectors.
If a corpus contains $|\mathbb{V}|$ words which are linearly independent, then every word represents an axis in the continuous vector space $\mathbb{R}$.
Each word takes an independent axis which is orthogonal to other words/axes.
Then $\mathbb{R}$ will contain $|\mathbb{V}|$ axes.

Let us assume that the words in a corpus are considered as linearly independent basis vectors.
If a corpus contains $|\mathbb{V}|$ words which are linearly independent, then every word represents an axis in the continuous vector space $\mathbb{R}$.
Each word takes an independent axis which is orthogonal to other words/axes.
Then $\mathbb{R}$ will contain $|\mathbb{V}|$ axes.

Examples

1. The vocabulary size of *emma corpus* is 7079. If we plot all the words in the real space $\mathscr{R}$, we get 7079 axes
2. The vocabulary size of *Google News Corpus corpus* is 3 million. If we plot all the words in the real space $\mathscr{R}$, we get 3 million axes

DOCUMENT VECTOR SPACE MODEL

- Vector space models are used to represent words in a continuous vector space $\mathscr{R}$

DOCUMENT VECTOR SPACE MODEL

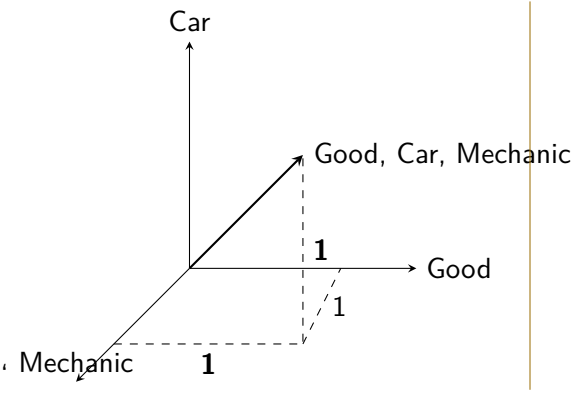
- Vector space models are used to represent words in a continuous vector space $\mathscr{R}$
- Combination of Terms represent a document vector in the word vector space

DOCUMENT VECTOR SPACE MODEL

- Vector space models are used to represent words in a continuous vector space $\mathscr{R}$
- Combination of Terms represent a document vector in the word vector space
- Very high dimensional space - several million axes, representing terms and several million documents containing several terms

EXAMPLE

Let us consider three words - *good, car, mechanic* and we will represent these words in a 3-D vector space



	Good	Car	Mechanic
D1	1	1	1
D2	1	0	1
D3	0	1	1

Term-term representation of words

DOCUMENT-TERM MATRIX

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12
t1	0.1	0.0	0.4	0.1	0.2	0.0	0.1	0.9	0.9	0.3	0.0	0.8
t2	0.1	0.0	0.4	0.1	0.2	0.0	0.1	0.9	0.9	0.3	0.0	0.8
t3	0.0	0.9	0.0	0.2	0.3	0.1	0.7	0.0	0.2	0.7	0.5	0.5
t4	0.0	0.9	0.3	0.9	0.5	0.1	0.9	0.3	0.8	0.4	0.1	0.4
t5	0.4	0.0	0.3	0.2	0.5	0.9	0.3	0.7	0.4	0.6	0.0	0.3
t6	0.6	0.0	0.4	0.7	0.3	0.3	0.9	0.1	0.9	0.0	0.0	0.3
t7	0.0	0.8	0.5	0.6	0.6	0.6	0.0	0.1	0.4	0.9	0.3	0.1
t8	0.4	0.0	0.6	0.5	0.5	0.1	0.7	0.1	0.5	0.3	0.8	0.1
t9	0.3	0.0	0.7	0.9	0.8	0.7	0.7	0.8	0.6	0.6	0.8	0.0
t10	0.0	0.5	0.5	0.0	0.2	0.0	0.0	0.1	0.3	0.4	0.5	0.3

The columns of the matrix represent the document as vectors. A document vector is represented by the terms present in the document

TERM-TERM MATRIX

	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12
t1	0.1	0.0	0.4	0.1	0.2	0.0	0.1	0.9	0.9	0.3	0.0	0.8
t2	0.1	0.0	0.4	0.1	0.2	0.0	0.1	0.9	0.9	0.3	0.0	0.8
t3	0.0	0.9	0.0	0.2	0.3	0.1	0.7	0.0	0.2	0.7	0.5	0.5
t4	0.0	0.9	0.3	0.9	0.5	0.1	0.9	0.3	0.8	0.4	0.1	0.4
t5	0.4	0.0	0.3	0.2	0.5	0.9	0.3	0.7	0.4	0.6	0.0	0.3
t6	0.6	0.0	0.4	0.7	0.3	0.3	0.9	0.1	0.9	0.0	0.0	0.3
t7	0.0	0.8	0.5	0.6	0.6	0.6	0.0	0.1	0.4	0.9	0.3	0.1
t8	0.4	0.0	0.6	0.5	0.5	0.1	0.7	0.1	0.5	0.3	0.8	0.1
t9	0.3	0.0	0.7	0.9	0.8	0.7	0.7	0.8	0.6	0.6	0.8	0.0
t10	0.0	0.5	0.5	0.0	0.2	0.0	0.0	0.1	0.3	0.4	0.5	0.3
t11	0.01	0.2	0.4	0.1	0.2	0.2	0.0	0.0	0.0	0.1	0.2	0.0
t12	0.1	0.12	0.54	0.01	0.02	0.0	0.0	0.0	0.0	0.6	0.7	0.0

The columns and rows of the matrix represent the words as vectors.

Do they represent the contextual relationship among words?

A similarity measure is a real-valued function that quantifies the similarity between two objects - in this case words Some of the similarity measures are given below.

$$\textbf{Euclidean Distance - } \mathscr{E}(\vec{w_1}, \vec{w_2}) = \sqrt{w_1^2 - w_2^2}$$

$$\textbf{Cosine Similarity} = \frac{\vec{w_1}.\vec{w_2}}{\|\vec{w_1}\| \|\vec{w_2}\|} = \frac{\vec{w_1}}{\|\vec{w_1}\|} \cdot \frac{\vec{w_2}}{\|\vec{w_2}\|}$$

$$\textbf{Cosine distance} = 1 - \frac{\vec{w_1}.\vec{w_2}}{\|\vec{w_1}\| \|\vec{w_2}\|} = \frac{\vec{w_1}}{\|\vec{w_1}\|} \cdot \frac{\vec{w_2}}{\|\vec{w_2}\|}$$

$$\textbf{Cluster similarity-} \mathscr{L}(\vec{w_1}, \vec{w_2}) = \frac{\vec{w_1}.\vec{w_2}}{\|\vec{w_1}\|}$$

Vector Semantics

VECTOR REPRESENTATION OF WORDS

Let V be the unique set of terms and $|V|$ be the size of the vocabulary. Then every vector representing the word $\mathscr{R}^{|V|x1}$ would point to a vector in the V-dimensional space

ONE-HOT VECTOR - 1

Consider all the ≈39000 words (estimated tokens in English is ≈ 13M) in the Oxford Learner's pocket dictionary. We can represent each word as an independent vector quantity as follows in the real space $\mathscr{R}^{|V|X1}$

$$t^{a} = \begin{pmatrix} 1 \\ 0 \\ \dots \\ 0 \\ \dots \\ 0 \\ 0 \end{pmatrix} \quad t^{aback} = \begin{pmatrix} 0 \\ 1 \\ \dots \\ 0 \\ \dots \\ 0 \\ 0 \end{pmatrix} \quad \dots \; t^{zoom} = \begin{pmatrix} 0 \\ 0 \\ \dots \\ 0 \\ \dots \\ 1 \\ 0 \end{pmatrix} \quad t^{zucchini} = \begin{pmatrix} 0 \\ 0 \\ \dots \\ 0 \\ \dots \\ 0 \\ 1 \end{pmatrix}$$

This is a very simple codification scheme to represent words independently in the vector space. This is known as ***one-hot vector***.

ONE-HOT VECTOR - 2

In one-hot vector, every word is represented independently. The terms, *home, house, apartments, flats* are independently coded. With one-hot vector based model, the dot product

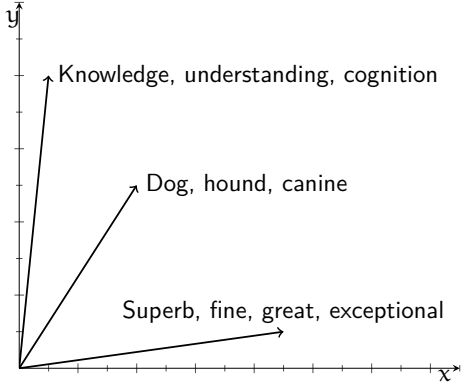
$$\left(t^{\text{House}}\right)^{T} \cdot t^{\text{Apartment}} = 0 \tag{8}$$

$$\left(t^{\text{Home}}\right)^{T} \cdot t^{\text{House}} = 0 \tag{9}$$

With one-Hot vector, there is no notion of similarity or synonyms.

RELATIONSHIP AMONG TERMS - SYNONYMS

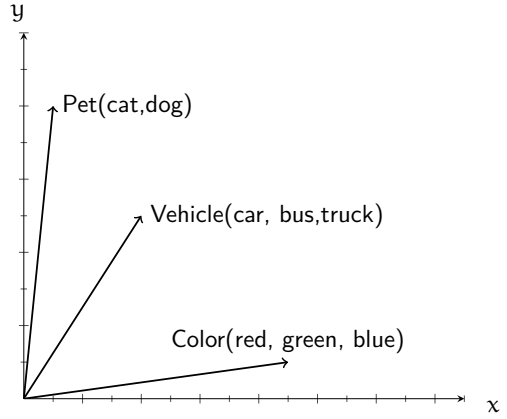
We could represent all the synonyms of a word in one axis



- Can we assume that similar words be situated/placed in the respective concept space?
- Can we assume that the properties of the word that inherit the properties of that space?

RELATIONSHIP AMONG TERMS - IS-A VECTOR

We could represent inheritance relationships of words as vectors.



SYNONYMS

small.a.01	['small', 'little']
minor.s.10	['minor', 'modest', 'small', 'small-scale', 'pocket-size', 'pocket-sized']
humble.s.01	['humble', 'low', 'lowly', 'modest', 'small']
little.s.07	['little', 'minuscule', 'small']
belittled.s.01	['belittled', 'diminished', 'small']
potent.a.03	['potent', 'strong', 'stiff']
impregnable.s.01	['impregnable', 'inviolable', 'secure', 'strong', 'unassailable', 'hard']
	He has such an impregnable defense (Cricket-Very hard to find the gap between the bat and the pad)
solid.s.07	['solid', 'strong', 'substantial']
strong.s.09	['strong', 'warm']
firm.s.03	['firm', 'strong'] - firm grasp of fundamentals

POLYSEMOUS WORD - BANK

Synset('bank.n.01')	sloping land (especially the slope beside a body of water)
Synset('depository-financial-institution.n.01')	a financial institution that accepts deposits and channels the money into lending activities
Synset('bank.n.03')	a long ridge or pile
Synset('bank.n.10')	a flight maneuver; aircraft tips laterally about its longitudinal axis (especially in turning)
Synset('trust.v.01')	have confidence or faith in

Bank appears in different word senses - or the meaning of the word is determined by the context in which appears

How do we place theses polysemous word in an axis? Using multiple vectors?

CONTEXTUAL UNDERSTANDING OF TEXT

You shall know a word by the company it keeps - (Firth, J. R. 1957)

- In order to understand the word and its meaning, it not enough if we consider only the individual word
- The *meaning* and *context* should be central in understanding word/text
- Exploit the context-dependent nature of words
- Language patterns cannot be accounted for in terms of a single entity
- The *collocation*, a particular word consistently co-occurs with the other words, gives enough clue to understand a word and its meaning

UNDERSTANDING A WORD FROM ITS CONTEXT

The view from the top of the mountain was
The view from the summit was
La vue du sommet de la montagne était
Mtazamo wa juu wa mlima huo ulikuwa

awesome/(*impressionnante*, *impressionnant*)
breathtaking
amazing, அற்புதமான/అద్భుతకరమాయ/
stunning/(*superbe*)
astounding अद्भुत/চমকপ্রদ
astonishing
awe-inspiring
extraordinary
incredible/(*incroyable*)
unbelievable
magnificent शानदार/ಗಂಭೀರಮಾಯ/ભવ્ય
wonderful/(*ajabu*)
spectacular
remarkable/(*yakuvutia*)

AJI AMARILLO

- Heard of Aji Amarillo?

AJI AMARILLO

- Heard of Aji Amarillo?
- It measures 30K/50K on the Scoville Heat scale

AJI AMARILLO

- Heard of Aji Amarillo?
- It measures 30K/50K on the Scoville Heat scale
- It is not as hot as bhüt jolokia which measures 1 million on the same scale

AJI AMARILLO

- Heard of Aji Amarillo?
- It measures 30K/50K on the Scoville Heat scale
- It is not as hot as bhüt jolokia which measures 1 million on the same scale
- If you visit Peru, do not miss dishes made from this

AJI AMARILLO

- Heard of Aji Amarillo?
- It measures 30K/50K on the Scoville Heat scale
- It is not as hot as bhüt jolokia which measures 1 million on the same scale
- If you visit Peru, do not miss dishes made from this
- They are yellow in color and have a balanced, fruity flavor close to mango and even passion fruit

AJI AMARILLO

- Heard of Aji Amarillo?
- It measures 30K/50K on the Scoville Heat scale
- It is not as hot as bhüt jolokia which measures 1 million on the same scale
- If you visit Peru, do not miss dishes made from this
- They are yellow in color and have a balanced, fruity flavor close to mango and even passion fruit
- It is a member of the capsicum baccatum

AJI AMARILLO

- Heard of Aji Amarillo?
- It measures 30K/50K on the Scoville Heat scale
- It is not as hot as bhüt jolokia which measures 1 million on the same scale
- If you visit Peru, do not miss dishes made from this
- They are yellow in color and have a balanced, fruity flavor close to mango and even passion fruit
- It is a member of the capsicum baccatum
- Star ingredient in many Peru's dishes

AJI AMARILLO

- Heard of Aji Amarillo?
- It measures 30K/50K on the Scoville Heat scale
- It is not as hot as bhüt jolokia which measures 1 million on the same scale
- If you visit Peru, do not miss dishes made from this
- They are yellow in color and have a balanced, fruity flavor close to mango and even passion fruit
- It is a member of the capsicum baccatum
- Star ingredient in many Peru's dishes

Inference

Aji Amarillo is a hot food item
Aji Amarillo is a vegetable
Aji Amarillo is grown in Peru

Aji Amarillo is a member of capsicum baccatum family

IDEAL PROPERTIES OF WORD VECTORS

- Encode the relationship among words
- Identify similar words using
  - Law of similarity - "**car**" and "**automobile**" vectors are close due to shared context
  - Law of contiguity - "**coffee**" and "**cup**" vectors show proximity because they occur together
  - Law of contrast - "**hot**" and "**cold**" vectors are distinct yet related as antonyms
  - Reduce word-vector space into a smaller sub-space
- Extract semantic information - capture words that demonstrates context, represents relationship
  New Delhi, India, capital - semantically connected through geography and semantics
- Represent polysemous words - multiple senses of a word based on different contexts

SEMANTICALLY CONNECTED VECTORS

- Identify a model that enumerates the relationships between terms
- Identify a model that tries to place similar items closer to each other in some space or structure
- Build a model that discovers/uncovers the semantic similarity between words and documents in the latent semantic domain
- Develop a distributed word vectors or dense vectors that captures the linear combination of word vectors in the transformed domain
- Similar to the term-document space identify a semantic space for similar words

WORD SIMILARITY

- Sparse vectors are too long and not very convenient as features machine learning
- Abstracts more than just frequency counts
- It captures neighborhood words that are connected by synonyms

METHODS TO CREATE WORD VECTORS

- Brown clustering - statistical algorithms for assigning words to classes based on the frequency of their co-occurrence with other words. It creates a binary tree of word clusters where words in the same cluster tend to appear in similar contexts, helping capture semantic relatedness
- Hyperspace Analogue to Language - HAL
- Correlated Occurrence Analogue to Lexical Semantic - COALS
- Latent Semantic Analysis or Latent Semantic Indexing
- Global Vectors - GloVe
- Neural networks using skip grams and CBOW
  - CBOW - uses surrounding words to predict the center of words
  - Skip grams use center of words to predict the surrounding words

You shall know a word by the company it keeps

- Firth, 1957

ATTRIBUTIONAL SIMILARITY

Attributional Similarity

- Two words are similar if they shared similar attributes - cat and kitten, dog and puppy
- Refers to the degree of similarity between two words or phrases in terms of their shared attributes
- Words that share many collocates denote concepts that share many attributes

Relational Similarity

- Related by concepts/roles - King and queen are related by the roles in the monarchy
- Blood relationships - siblings, aunts, uncles, parents, etc.

Techniques to extract similarities - Distributional Semantic Models

VECTOR BASED MODELS

Assumption Context words within a certain distance from the target word are semantically relevant

- Ability to represent word meaning simply by using distributional statistics
- The context surrounding a given word provides important information about its meaning
  - Small number of words surrounding the target word is known as context
- Distributional patterns of co-occurrence with their neighboring words provide semantic properties of words

A SAMPLE CO-OCCURRENCE MATRIX WITH RAW FREQUENCY COUNT

	w_0	w_1	w_2	...	w_{n-3}	w_{n-2}	w_{n-1}	w_n
w_0	0	33	29	...	33	37	39	39
w_1	33	1	45	...	0	27	21	10
w_2	29	45	0	...	37	40	19	23
...	...	...	...	...	...	...	...	...
w_{n-3}	33	0	37	...	0	24	26	49
w_{n-2}	37	27	40	...	24	0	22	31
w_{n-1}	39	21	19	...	26	22	1	38
w_n	39	10	23	...	49	31	38	0

- This matrix captures the lexical space of the corpus
- Statistical knowledge about words and their relationship in terms of co-occurrence are static in nature

TECHNIQUES TO CAPTURE CO-OCCURRENCE INFORMATION

Let A denote the word-word co-occurrence matrix where each row corresponds to a unique/target word, and each column represents a context.
a_{ij} denotes every element of the A

a_i denotes the number of times the word i co-occurring with the word j, $a_i = \sum_j a_{ij}$

Now, we can fill the co-occurrence using:

1. **Raw Frequency Count:** Each element, a_{ij} denotes the raw frequency count of word i co-occurring with the word j
2. **Probability:** $p_{ij} = P(w_j \mid w_i) = \frac{a_{ij}}{a_i}$
3. **Positive Point-wise Mutual Information(PMI)**:
   $PPMI(w_i, w_j) = max\left(\log_2\left(\frac{p_{ij}}{p_i * p_j}\right), 0\right)$

The co-occurrence statistics are captured by scaning the entire corpus once using a windowing approach

IMPORTANCE OF PPMI IN WORD SEMANTIC RELATIONSHIPS

Quantifying True Semantic Association

Measures how much more often two words co-occur than expected by random chance, capturing strong contextual links (e.g., “ice” and “cold”).

Correcting Frequency Bias

Standard PMI overemphasizes rare words that appear together just once by chance. PPMI filters this uninformative, low-frequency background noise.

Eliminating Negative Noise

By replacing negative scores with $\mathbf{0}$ via $\max(0, \text{PMI})$, it yields highly accurate, zero-grounded semantic vector spaces ideal for similarity tasks.

EXAMPLE: COMPUTING PPMI WITH LAPLACE SMOOTHING

Corpus Setup: Total words $N = 1{,}000{,}000$. Smoothed Context Weight $\alpha = 0.75$.
Smoothed context counts: $c_\alpha(w) = N \times \left(\frac{c(w)}{N}\right)^\alpha$.

Frequent Pair: (data, science)

$c_{data} = 10{,}000$, $c_{science} = 1{,}000$
Raw Joint Count $= 100$

$$P_\alpha(\text{science}) \approx \frac{1{,}000^{0.75}}{1{,}000^{0.75} + \ldots} \approx 0.0031$$

$$P(\text{data}) = 0.01$$

$$P(\text{joint}) = 0.0001$$

$$\text{PMI}_\alpha = \log_2\left(\frac{0.0001}{0.01 \times 0.0031}\right) = 1.69$$

$$\text{PPMI} = \max(0, 1.69) = 1.69$$

Rare Pair: (data, cryptozoology)

$c_{data} = 10{,}000$, $c_{crypto} = 2$
Raw Joint Count $= 1$ (Fluke)

$$P_\alpha(\text{crypto}) \approx \frac{2^{0.75}}{1{,}000^{0.75} + \ldots} \approx 0.00003$$

$$P(\text{data}) = 0.01$$

$$P(\text{joint}) = 0.000001$$

$$\text{PMI}_\alpha = \log_2\left(\frac{0.000001}{0.01 \times 0.00003}\right) = -1.58$$

$$\text{PPMI} = \max(0, -1.58) = 0$$

Conclusion: Raising rare counts to $\alpha = 0.75$ inflates their expected baseline probability, forcing the PMI due to uncommon occurrence to negative so PPMI safely squashes it to 0.

A similarity measure is a real-valued function that quantifies the similarity between two objects - in this case words. Some of the similarity measures are given below.

$$\textbf{Euclidean Distance} - \mathbb{E}(\vec{w_1}, \vec{w_2}) = \sqrt{w_1^2 - w_2^2} \tag{10}$$

$$\underset{\in[-1,1]}{\textbf{Cosine Similarity}} = \frac{\vec{w_1}.\vec{w_2}}{\|\vec{w_1}\| \, \|\vec{w_2}\|} \tag{11}$$

$$\underset{\in[0,1]}{\textbf{Cosine Distance}} = 1 - \textbf{Cosine Similarity} \tag{12}$$

WORD VECTOR EXAMPLES

Similar words for $\overrightarrow{\text{apple}}$

Term	Score
apple	0.000
iphone	0.266
ipad	0.287
apples	0.356
blackberry	0.361
ipod	0.365
macbook	0.383
mac	0.391
android	0.391
google	0.395
microsoft	0.418
ios	0.433
iphones	0.445
touch	0.446
sony	0.447

Similar words for $\overrightarrow{\text{american}}$

Word	Score
american	0.000
america	0.255
americans	0.312
u.s.	0.320
british	0.323
canadian	0.329
history	0.356
national	0.364
african	0.374
society	0.375
states	0.386
european	0.387
world	0.394
nation	0.399
us	0.399

VECTOR DIFFERENCE BETWEEN TWO WORDS

Table: Word Similarity Scores

$\overrightarrow{\text{Word}}$	$\overrightarrow{\text{Score}}$
raisin	0.5745
pecan	0.5761
cranberry	0.5840
butternut	0.5882
cider	0.5911
apricot	0.6037
tomato	0.6074
rosemary	0.6151
rhubarb	0.6158
feta	0.6183
apples	0.6226
avocado	0.6235
fennel	0.6306
chutney	0.6313
spiced	0.6328

VECTOR ARITHMETIC ON WORD VECTORS...

840B words and 300 elements word vectors used for this computation

$\overrightarrow{\text{apple}}$	$\overrightarrow{\text{apple}} - \overrightarrow{\text{iphone}}$	$\overrightarrow{\text{apple}} - \overrightarrow{\text{fruit}}$
('apple', 0)	('apples', 0.39)	('ipad', 0.412)
('apples', 0.25)	('fruit', 0.43)	('iphone', 0.433)
('blackberry', 0.31)	('grape', 0.44)	('macbook', 0.435)
('Apple', 0.35)	('tomato', 0.44)	('ipod', 0.445))
('iphone', 0.37)	('pecan', 0.45)	('imac', 0.465
('fruit', 0.37)	('rhubarb', 0.45)	('3gs', 0.473)
('blueberry', 0.38)	('pears', 0.45)	('Ipad', 0.490)
('strawberry', 0.38)	('cranberry', 0.452)	('itouch', 0.512)
('ipad', 0.39)	('raisin', 0.453)	('ipad2', 0.514)
('pineapple', 0.39)	('apricot', 0.459)	('Iphone', 0.514)
('pear', 0.39)	('carrot', 0.461)	('ios', 0.520)
('cider', 0.39)	('candied', 0.462)	('Macbook', 0.524)
('mango', 0.40)	('blueberry', 0.463)	('ibook', 0.534)
('ipod', 0.40)	('apricots', 0.466)	('IPhone', 0.541)
('raspberry', 0.40)	('tomatoes', 0.466)	('32gb', 0.545)