Word Representation

From Counts to Dense Vectors

Ramaseshan Ramachandran

Word Semantic Space

WHAT IS CONTEXT?

The main challenge in understanding how words relate to real-world situations has increasingly become a matter of context.

- All words within a window or ideally within a sentence
- All content words within a window or sentence that fall in a certain frequency range
- All content words which stand in closest proximity to the word in question in the grammatical schema of each window or sentence

CONTEXT - THE INFLUENCER

Phrase	Meaning
Spill the tea	Share the gossip or details
On the same page	Everyone understanding or agreeing on something.
I'm so over it	Tired of something
That's so cool	Approval or exclamation
It's raining cats and dogs	It's raining heavily

- Context influences the word meaning
- Small boy, small car, small house, small island
- Words that occur in similar contexts will tend to have similar meanings
- Semantic similarity between two words (w_x, w_y) is a function of how frequently they appear in similar linguistic contexts
  - $\vec{w_x} \approx \vec{w_y}$ when the frequency of the context $(f_{C_{xy}(k)})$ with a window of size k in which both words w_x and w_y appeared is higher
- If $f_{C_{xy}}(k)$ is higher, then the semantic relationship of (w_x, w_y) is stronger
- Extending to multiple similar words for w_x:

  $\vec{w_x} \approx \vec{w_{y_i}}$ when the frequency of $C_{xy_i}(k)$ is higher, where $i = 1 \ldots n$

  Note: Here $\approx$ represents similarity

SEMANTIC SPACE OR CONCEPT SPACE

Hyponyms and hypernyms are linguistic terms that describe the relationship between words based on their meaning.

- A space where similar words (synonyms, hyponyms, hypernyms) are classified and arranged along various axes
  - Colour (hypernym) - $\underbrace{\text{red}, \text{Green}, \text{Orange}}_{\text{co-hyponyms}}$ (hyponyms) - Attributional Similarity
- Models that automatically find similar words are known as Distributional Semantic Models (DSM)
- Semantically similar words are found automatically using co-occurrences/co-locations/context

  OR
- Words connected by similar patterns are probably semantically similar

Side-effect: Lexical co-occurrences associate semantically dis-similar words

LAWS OF ASSOCIATION

- **The law of similarity**: If two things are similar, the thought of one will tend to trigger the thought of the other
- **The law of frequency**: The more often two things or events are linked, the more powerful will be that association.
- **The law of contiguity**: Things or events that occur close to each other in space or time tend to get linked together in the mind.
- **The law of contrast**: Seeing or recalling something may also trigger the recollection of something completely opposite.

DISTRIBUTED SEMANTIC MODELS

- Extract the meaning of the words using distributed linguistic properties
- Compute lexical **co-occurrence** of every word (co-locates with certain distance) with every other word in the Vocabulary
  - Linear proximity of words within a window is considered
  - They need not represent any relations

  Example He drove the car through a red bridge.
  The verb drove relates to red and bridge only through the proximity,
  but carries no relations with red and bridge in terms of semantics
- Build a co-occurrence matrix using co-occurrence statistics
- Rows/columns in the matrix represent distributed semantic information of words

Distributed Semantic Models

DISTRIBUTED SEMANTIC MODELS

Context is essential in the mapping the relationship between words and concepts.

I cook dinner every Sunday
...
I cooked dinner last Sunday
...
I am cooking dinner today
...
My son cooks dinner every Sunday
...

- The words in this corpus are related by association
- The verb forms cook, cooked, cooks, and cooking are related through their co-occurrence statistics - a semantic relationship
- The words dinner and Sunday are linked by an associative relationship arising from co-occurrence

- In the COVID19 research corpus, one may not find the phrase *needle in a haystack*
- You will find the word needle will be lexically associated with $\underbrace{\texttt{pain}, \texttt{illness}, \texttt{blood}, \texttt{drugs}, \texttt{syringe}}_{\text{Associative relationship}}$, but not to thread, knitting, cloth, etc.

You shall know a word by the company it keeps

- Firth, 1957

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

---

Write $tf_{t,d}$ for the number of times term t occurs in document d, and M_d for the total number of tokens in d. Those two symbols carry the rest of this session.

TERM-FREQUENCY WEIGHTING SCHEMES

$$\text{Boolean} \quad 0 \text{ or } 1 \tag{1}$$

$$\text{Raw count} \quad \mathrm{tf}_{t,d} \tag{2}$$

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

$$\text{Log weighting} \quad \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d} & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

- **Boolean** throws away the count. This is the incidence matrix
- **Raw count** keeps everything, including the fact that long documents have larger counts for no interesting reason
- **Length adjustment** turns the count into a rate, so a 200-word abstract and a 20,000-word thesis become comparable
- **Log weighting** keeps the ordering but flattens the scale. The $+1$ makes a single occurrence score 1 rather than 0

WHY DAMP THE COUNT AT ALL?

Relevance does not grow linearly with occurrences. A document that says *cricket* twenty times is more about cricket than one that says it twice. It is not ten times more about cricket.

$\mathrm{tf}_{t,d}$	raw	tf/M_d ($\mathrm{M}_d = 1000$)	$1+\log_{10}\mathrm{tf}$
1	1	0.001	1.000
2	2	0.002	1.301
10	10	0.010	2.000
100	100	0.100	3.000
1000	1000	1.000	4.000

- A thousandfold spread in raw counts becomes a fourfold spread in weight
- The logarithm turns *multiplicative* differences into *additive* ones, which is the same trick that made Zipf's law a straight line
- Length adjustment and log damping fix different problems and are often used together

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 the highest score in almost every document
- Zipf's law guarantees this. The largest counts in any document belong to the least informative words
- Classification and retrieval both suffer when the score is dominated by words shared across the whole collection

---

Term frequency measures importance within a document. Nothing so far measures importance across the collection.

INVERSE DOCUMENT FREQUENCY

We want to scale down the weight of terms that occur everywhere and boost the weight of terms that occur in few places.
Let df_t be the **document frequency**, the number of documents that contain t at least once, out of N documents in the collection. Then

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

- df_t counts *documents*, not occurrences. A term used 500 times in one document still has $\mathrm{df}_t = 1$
- $\mathrm{df}_t \leq N$ always, so the ratio is at least 1 and the idf is never negative
- If a term appears in every document, $\mathrm{df}_t = N$, so $\mathrm{idf}_t = \log 1 = 0$. The term cannot discriminate, and its weight is annihilated
- The rarer the term, the larger the weight. It is a measure of **informativeness**

READING THE EQUATION: WHY A LOGARITHM?

Pick a document at random. The chance it contains t is $P(t) = df_t/N$. Then

$$idf_t = \log \frac{N}{df_t} = -\log P(t) \tag{6}$$

which is exactly the **self-information** of the event "this document contains t" [**Manning2009**].

- idf is not an arbitrary formula. It is the number of bits of surprise in seeing the term, measured in whatever base the log uses
- A term you expected to see carries no information. A term you did not expect carries a lot
- The logarithm also tames the scale. Without it, N/df_t ranges over five orders of magnitude and one rare word would dominate every score
- The base only rescales every weight by a constant, so it never changes a ranking. Use $\log_{10}$ by hand and $\log_e$ in code

TWO PRACTICAL VARIANTS

$$\text{smoothed} \quad \mathrm{idf}_t = \log\left(\frac{N}{1+\mathrm{df}_t}\right)+1 \tag{7}$$

$$\text{probabilistic} \quad \mathrm{idf}_t = \log\left(\frac{N-\mathrm{df}_t}{\mathrm{df}_t}\right) \tag{8}$$

- The $1+\mathrm{df}_t$ guards against a query term that appears in *no* document, where the plain formula divides by zero. The trailing $+1$ stops a term in every document from collapsing the whole vector to zero
- `scikit-learn` uses the smoothed form by default, so its numbers will not match a hand computation unless you say so
- The probabilistic form goes negative for terms in more than half the documents, and is the ancestor of the **BM25** weighting we meet in session 21
- All three agree on the ordering of terms. They differ in how hard they punish the head

TF-IDF

Multiplying the two factors gives a composite weight for each term in each document

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t \tag{9}$$

- $\text{tf}_{t,d}$ asks: is this term important *inside this document*?
- idf_t asks: is this term informative *across the collection*?
- The product is high only when both are high, which is exactly the profile of a word that characterises the document
- It is very low when a term appears in every document, whatever its count

---

At bottom, idf is a machine for undoing Zipf's law. It discounts the frequent words that the frequency distribution guarantees you will have.

WORKED EXAMPLE: ONE TERM IN A LARGE COLLECTION

IDF measures how rare the term t is across the collection:

$$\mathrm{idf}_t = \log_{10}\left(\frac{\text{total documents in the corpus}}{\text{documents containing t}}\right)$$

Take N = 100,000 documents. The word ***moon*** occurs in 100 of them, so

$$\mathrm{idf}(\textit{moon}) = \log_{10}\frac{100{,}000}{100} = \log_{10} 1000 = 3$$

In a 427-token document, *moon* appears 20 times:

$$\mathrm{tf} = \frac{20}{427} = 0.047 \qquad \text{tf-idf} = 0.047 \times 3 = \boxed{\mathbf{0.141}}$$

Now ***the***, in all 100,000 documents and 45 times in this one:

$$\mathrm{idf}(\textit{the}) = \log_{10} 1 = 0 \qquad \text{tf-idf} = \frac{45}{427} \times 0 = \boxed{\mathbf{0}}$$

The most frequent word in the language scores zero, whatever its count.

The same term *moon* in five documents of that collection. Rank by tf-idf:

Document	tf	tf-idf	Rank
d_1	$20/427 = 0.047$	0.14	3
d_2	$30/250 = 0.120$	0.36	1
d_3	$20/250 = 0.080$	0.24	2
d_9	$5/125 = 0.040$	0.12	4
d_{1000}	$20/1000 = 0.020$	0.06	5

Every document shares the same $\mathrm{idf} = 3$, so within one query term the ranking is decided by tf alone. Note d_1 and d_3: the same raw count of 20, different lengths, different ranks.

WORKED EXAMPLE, STAGE 1: COUNT

A collection of $N = 4$ documents.

d_1: `the moon orbits the earth` d_2: `the earth orbits the sun`
d_3: `the telescope shows the moon` d_4: `the sun is a star`

Rows are terms, columns are documents. Each cell is $tf_{t,d}$.

term	d_1	d_2	d_3	d_4
the	2	2	2	1
moon	1	0	1	0
orbits	1	1	0	0
earth	1	1	0	0
sun	0	1	0	1
telescope	0	0	1	0
shows	0	0	1	0
is	0	0	0	1
a	0	0	0	1
star	0	0	0	1

By raw count, *the* is the most important word in all four documents.

WORKED EXAMPLE, STAGE 2: COUNT DOCUMENTS

df_t counts documents, not occurrences.

term	df_t	N/df_t	$\mathrm{idf}_t = \log_{10}(\mathrm{N}/\mathrm{df}_t)$
the	4	1.00	**0.000**
moon	2	2.00	0.301
orbits	2	2.00	0.301
earth	2	2.00	0.301
sun	2	2.00	0.301
telescope	1	4.00	0.602
shows	1	4.00	0.602
is	1	4.00	0.602
a	1	4.00	0.602
star	1	4.00	0.602

- *the* is in every document, so $\mathrm{N}/\mathrm{df} = 1$ and its weight is 0
- Here *a* has $\mathrm{df} = 1$ and scores the *highest* idf. idf is a statistic of the collection, not a judgement about language, and with $\mathrm{N} = 4$ it measures almost nothing

WORKED EXAMPLE, STAGE 3: MULTIPLY

$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$, cell by cell.

term	d_1	d_2	d_3	d_4
the	0.000	0.000	0.000	0.000
moon	0.301	0.000	0.301	0.000
orbits	0.301	0.301	0.000	0.000
earth	0.301	0.301	0.000	0.000
sun	0.000	0.301	0.000	0.301
telescope	0.000	0.000	0.602	0.000
shows	0.000	0.000	0.602	0.000
is	0.000	0.000	0.000	0.602
a	0.000	0.000	0.000	0.602
star	0.000	0.000	0.000	0.602
$\|\vec{d}\|$	0.5214	0.5214	0.9031	1.0854

The whole first row is zero. **Two documents that share only *the* now share nothing at all.**

WORKED EXAMPLE, STAGE 4: RANK FOR A QUERY

Score a document by adding the tf-idf of the query terms:

$$\text{score}(q, d) = \sum_{t \in q} \text{tf-idf}_{t,d} \tag{10}$$

query *"the moon"*

doc	raw count	tf-idf
d_1	3	**0.3010**
d_3	3	**0.3010**
d_2	2	0.0000
d_4	1	0.0000

query *"moon orbits"*

doc	raw count	tf-idf
d_1	2	**0.6021**
d_2	1	0.3010
d_3	1	0.3010
d_4	0	0.0000

- On *"the moon"*, raw counts put d_2 third with a score of 2, ahead of d_4. Neither mentions the moon. tf-idf sends both to exactly zero
- The word *the* contributed nothing to either query, so the user was free to type it

WHY TF-IDF WHEN LLM CAN DO A LOT?

- **Hybrid Search Standard:** Modern RAG pipelines and vector databases combine dense embeddings with sparse TF-IDF/BM25 for exact keyword matching.
- **Precision on Rare Terms:** Excels at out-of-vocabulary words, specialized technical jargon, product IDs, and code snippets.
- **High Efficiency:** Ultra-fast CPU execution with zero GPU requirements and minimal latency overhead.
- **Interpretability & Baselines:** Provides transparent feature weights and acts as a fast, reliable baseline model.
- **Pedagogical Foundation:** Essential for introducing core NLP concepts like document frequency weighting and vector space models.

OKAPI BM25 RANKING ALGORITHM

Relevance Score for Query $Q = \{q_1, \ldots, q_n\}$ and Document D:

$$\text{Score}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

Robertson-Spärck Jones IDF Variant

$$\text{IDF}(q_i) = \ln\left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1\right)$$

- $f(q_i, D)$: Raw term frequency of term q_i in document D.
- N & $n(q_i)$: Total documents in collection and number of docs containing q_i.
- $\text{avgdl} = \frac{\sum_{d \in D} |d|}{N}$. What is the importance of $\frac{|D|}{\text{avgdl}}$
- $|D|/\text{avgdl}$: Ratio of document length to average collection length.
- k_1 (default $\approx 1.2 - 2.0$): Non-linear term frequency saturation parameter.
- b (default ≈ 0.75): Document length normalization weighting parameter.

BAG OF WORDS

Representing a document by its term weights alone, $[w_1, w_2, w_3, \ldots, w_n]$, is the *bag-of-words* model

- Word order is ignored. Only the weights matter, so *"dog bites man"* and *"man bites dog"* get the same vector
- Two documents with similar bags are assumed to be similar in content
- Every term is its own dimension, orthogonal to every other. So *cat* and *dog* are exactly as unrelated as *cat* and *the*
- That last point is the flaw the rest of the course repairs. tf-idf weights the axes; it does not make similar words share them

---

Contrast: transformers restore word order through position information, and embeddings give related words nearby directions.

Empirical Laws

WHERE THIS SESSION SITS

Before any model, count the words. It is the cheapest experiment in NLP and one of the most consequential.

- **Zipf's law** says how frequency falls with rank
- **Heaps' law** says how the vocabulary grows with the corpus
- **Type–token ratio** tries to summarise lexical richness, and fails in an instructive way

---

Every later decision in this course rests on these three curves. Stop lists, tf-idf weighting, smoothing, subword vocabularies and even the size of an LLM tokenizer are all responses to them.

TWO WORDS WE MUST NOT CONFUSE

- A **token** is a running occurrence of a word
- A **type** is a distinct word

"the cat saw the dog" has **5 tokens** and **4 types**, because *the* occurs twice.

Write N for the number of tokens and V for the number of types. Everything in this session is a statement about how N and V relate.

Brown corpus, measured with the **Assignment 1 tokenizer**: $N = 1{,}023{,}734$ tokens and $V = 44{,}324$ types. The word *the* alone is 6.84% of all tokens. The top ten words are 24.1%. And 39.5% of the types occur exactly once.

Change the tokenizer and every one of those numbers changes. Always say which one you used.

ZIPF'S LAW

This empirical law models the frequency distribution of words across languages. Zipf's law [1] states that, in a given corpus, the frequency of a word is inversely proportional to its rank in the term-frequency table.

$$f(r) = \frac{C}{r^{\alpha}}, \qquad \alpha \approx 1 \tag{11}$$

- r is the **rank** of a word, so the most frequent word has $r = 1$
- $f(r)$ is the **frequency** of the word at rank r
- α is the **decay exponent**, which controls how fast frequency falls
- C is a constant of the corpus. Setting $r = 1$ shows that $C = f(1)$, the frequency of the most common word

With $\alpha = 1$ the second word is half as frequent as the first, the tenth is a tenth as frequent, and the hundredth a hundredth.

READING THE EQUATION: A POWER LAW IS A STRAIGHT LINE

Take $\log_{10}$ of both sides of $f(r) = C\, r^{-\alpha}$:

$$\log f(r) = \log C - \alpha \log r \tag{12}$$

- This is $y = mx + c$ with $y = \log f$, $x = \log r$, slope $m = -\alpha$ and intercept $c = \log C$
- So plot frequency against rank on **log–log axes** and the law becomes a check you can do by eye
- If the points lie on a line, the law holds. The steepness of the line is α
- Fitting α is now ordinary least squares on the logged data, not curve fitting

---

Why this matters

Every empirical law in this session is a power law, and every power law becomes a line under logarithms. Learn the trick once and you can fit Zipf, Heaps, neural scaling laws and retrieval latency curves the same way.

ZIPF'S LAW: FREQUENCY VS RANK

Output generated from Indian budget speeches using the bash script [44] with minimal preprocessing

Word	Frequency	Rank
the	59042	1
of	46110	2
to	35873	3
and	24661	4
in	22916	5
for	14426	6
a	14366	7
is	11155	8
be	9760	9
I	9070	10
...	...	...

Word	Frequency	Rank
agricultural	479	233
important	478	234
policy	477	235
reduced	476	236
balance	476	237
...	...	...
(SFB)	1	92363
(SFAC),1	1	92364
(SFAC)	1	92365
(SEZs).,1	1	92366
...	...	...

We now fit α to exactly these fifteen numbers.

WORKED EXAMPLE, STAGE 1: THE TEMPTING SHORTCUT

Two points are enough to solve for α, so try that first. Use ranks 1 and 10:

$$\frac{f(1)}{f(10)} = \frac{C/1^{\alpha}}{C/10^{\alpha}} = 10^{\alpha} \quad \Longrightarrow \quad \alpha = \log_{10}\frac{f(1)}{f(10)}$$

$$\alpha = \log_{10}\frac{59042}{9070} = \log_{10} 6.509 = \boxed{\mathbf{0.8136}}$$

Now test it. Predict the frequency at rank 235, where the table says 477:

$$\hat{f}(235) = \frac{59042}{235^{0.8136}} = \frac{59042}{84.9} = 695$$

Predicted 695, actual 477. The estimate is 46% too high.

WORKED EXAMPLE, STAGE 2: USE EVERY POINT YOU HAVE

Least squares on $\log f$ against $\log r$, over all fifteen rows:

$$\hat{\alpha} = -\frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}, \qquad x_i = \log r_i,\ y_i = \log f_i \tag{13}$$

Method	α	C	$\hat{f}(235)$
Two points, $r = 1$ and $r = 10$	0.8136	59,042	695 (+46%)
Least squares, all 15 points	**0.9426**	**83,239**	**485** (+2%)

- The shortcut discarded thirteen measurements and paid for it
- Note that $C = 83{,}239$ is larger than the observed $f(1) = 59{,}042$. The fitted line does not pass through the top word, and it should not
- The head of the distribution is flatter than the body, so any α read off the head alone comes out too small

WORKED EXAMPLE, STAGE 3: α IS NOT A CONSTANT

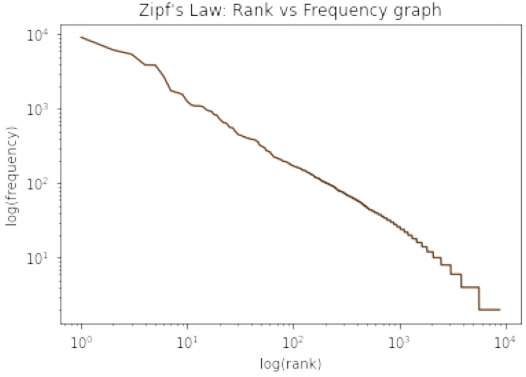
The same corpus gives a different α depending on the rank range you fit. Brown corpus, measured:

Rank range fitted	Fitted α
1…100	0.9816
1…1,000	0.9668
1…10,000	1.0726
1…44,324 (all types)	1.3492

- The tail falls faster than the head, so a wider window gives a steeper line
- An exponent quoted without its fitting range is not a measurement
- This is why the textbook claim "$\alpha \approx 1$" is safe. It is a statement about the first few thousand ranks

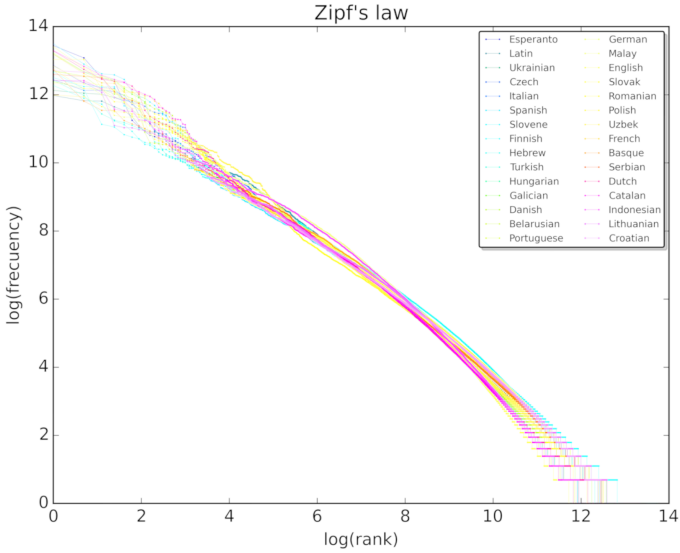
Always report the range you fitted over.

ZIPF'S LAW I



$$f \propto \frac{1}{r^{\alpha}}$$

A plot of the rank versus frequency in a log-log scale.



$$f \propto \frac{1}{r^{\alpha}}$$

A plot of the rank versus frequency for the first 10 million words in 30 Wikipedia (dumps from October 2015) in a log-log scale.

Source: Zipf's law -Wikipedia

FROM PROPORTIONALITY TO PROBABILITY

$f(r) = Cr^{-\alpha}$ gives counts. Divide by the total count to get a distribution over the vocabulary:

$$p(r) = \frac{r^{-\alpha}}{H_{V,\alpha}}, \qquad H_{V,\alpha} = \sum_{k=1}^{V} k^{-\alpha} \tag{14}$$

$H_{V,\alpha}$ is the **generalised harmonic number**. It is the normalising constant that makes the probabilities sum to one.

- $p(r)$ is the chance that a token drawn at random from the corpus is the rank-r word
- The **coverage** of the top k words follows immediately:

$$\text{Coverage}(k) = \sum_{r=1}^{k} p(r) = \frac{H_{k,\alpha}}{H_{V,\alpha}} \tag{15}$$

- For $\alpha \leq 1$ the sum grows without bound as V grows, which is exactly why the tail never becomes negligible

WORKED EXAMPLE: HOW MUCH TEXT DOES THE HEAD CARRY?

Brown corpus, $V = 44{,}324$, using $\alpha = 0.9668$ fitted on ranks $1 \ldots 1000$.

Top k types	% of the vocabulary	Coverage, model	Coverage, measured
10	0.02%	22.51%	24.09%
100	0.23%	41.40%	47.18%
1,000	2.26%	62.14%	69.13%
10,000	22.56%	84.56%	92.60%

- Read column two against column four. About 2% of the vocabulary carries about 69% of the running text
- The model tracks the measurement at the head and drifts at the tail, because one α cannot bend twice
- Dropping the rarest 77% of types costs you under 8% of the tokens

WHAT ZIPF'S LAW BUYS US IN THE NLP PIPELINE

- **Stop lists are short.** A few dozen words cover a quarter of every corpus, so removing them is cheap and changes the token count sharply
- **Raw counts are the wrong features.** The largest counts in a document belong to the least informative words. This is the whole motivation for inverse document frequency in the next session
- **Vocabulary truncation is nearly free in tokens.** Coverage tells you what a cut-off costs before you make it
- **Half the vocabulary is unreliable.** With 39.5% of types seen once and 54.3% seen twice or fewer, no statistic estimated per word can be trusted in the tail. This is why language models need smoothing
- **Subsampling frequent words.** word2vec discards frequent tokens with probability tied to their frequency, purely to undo the Zipfian head
- **Subword vocabularies.** BPE spends a fixed budget of merges on the head and composes the tail from pieces

Web Traffic Patterns

- A handful of sites take a large share of all visits
- The top 20% of websites account for roughly 80% of traffic
- The same shape as the Pareto principle

Caching Strategies

- CDNs exploit Zipf's law to decide what to keep hot
- A small number of pages receive most requests
- Coverage is exactly the cache hit rate you can expect

Search Algorithms

- Most queries concentrate on a small set of popular topics

Text Analytics

- Identifies stop words without a hand-written list
- Identifies the words worth focusing on
- Reduces the sample space

BASH SCRIPT FOR COMPUTING RANK AND FREQUENCY

```
# 10. Concatenate all files in TXT directory
# 11. Remove form feed character (^L)
# 12. Split into words (replace spaces with newlines)
# 13. Sort in reverse
# 14. Count unique words
# 15. Sort by frequency, descending
# 16. Add word, frequency rank
# 17. output as word_freq_rank.csv

cat xx.txt | \
tr -d '\f' | \
tr -s ' ' '\n' | \
sort -r | \
uniq -c | \
sort -nr | \
awk '{print $2 "," $1 "," NR }' \
> word_freq_rank.csv
```

MANDELBROT'S CORRECTION

Mandelbrot fits the observed distribution more closely by shifting the rank before the power is taken

$$f(r) = \frac{C}{(r+\beta)^{\alpha}} \tag{16}$$

- β moves every rank to the right, so the curve flattens where r is small and is left almost untouched where r is large
- It is a **head correction** and nothing else. Plain Zipf overestimates the very frequent words
- β sits inside the logarithm, so it cannot be found by least squares. Scan β, and least-squares α and C at each value
- How large β comes out depends almost entirely on **how much of the tail you fit**, as the next frame shows

WORKED EXAMPLE: DOES β EARN ITS PLACE?

Same fifteen budget-speech rows, same procedure, one extra parameter.

Model	α	β	Squared error	$\hat{f}(235)$
Zipf	0.9426	0	0.0453	485
Mandelbrot	0.9843	0.5	**0.0302**	**472**

Actual $f(235) = 477$. The error drops by a third for one extra parameter.

The warning that comes with it

The valley is shallow, so many pairs fit almost as well:

β	best α	squared error
0.0	0.9426	0.0453
0.5	0.9843	0.0302
1.0	1.0183	0.0356
2.0	1.0749	0.0643

Report (α, β) as a pair. Neither number means anything alone.

WORKED EXAMPLE: β IS SET BY THE FITTING RANGE

The budget example gave $\beta = 0.5$, which looks negligible. Fit Brown instead, and vary how much of the tail you include:

Rank range fitted	α	β
1…1,000	0.9668	**0.00**
1…10,000	1.0984	10.25
1…44,324 (all ranks)	1.5180	**403.25**

- Fit the head alone and β is exactly zero, so Mandelbrot buys nothing. Plain Zipf already describes the head
- Fit every rank, which is what the Assignment 1 grader does, and β runs into the hundreds
- A large β is not a red flag about your corpus. It is what a single power law needs in order to cover a curve that *bends*
- α and β rise together, because both are absorbing the same bend

Quote α, β and the rank range. All three, or none.

TYPE–TOKEN RATIO (TTR)

A natural summary of lexical richness is the ratio of types to tokens:

$$\text{TTR} = \frac{V}{N} = \frac{\text{number of distinct words}}{\text{total number of words}} \tag{17}$$

- TTR lies in $(0, 1]$. A text that never repeats a word scores 1
- Higher TTR is read as a richer vocabulary, and lower as repetitive or formulaic text
- *"the cat saw the dog"* gives $\text{TTR} = 4/5 = 0.8$
- Brown corpus as a whole gives $44{,}324/1{,}023{,}734 = 0.0433$

Those two numbers are not comparable, and the reason is the point of this section.

HEAPS' LAW TELLS US TTR MUST FALL

Substitute $V = KN^b$ into $TTR = V/N$:

$$TTR(N) = \frac{KN^b}{N} = KN^{b-1} \tag{18}$$

Since $b < 1$, the exponent $b - 1$ is negative, so TTR **falls as the text grows**. Brown's fit, $k = 16.18$ and $b = 0.5764$:

N	predicted TTR	measured TTR
1,000	0.8673	0.4680
10,000	0.3270	0.2584
100,000	0.1233	0.1296
1,000,000	0.0465	0.0436

Same author, same style, four different answers.

TTR is not a property of a text. It is a property of a text *at a length*.

Note the first row. The fit is good near a million tokens and poor at a thousand, because the curve is concave and 1,024 points pull the line towards the large-N end. A fitted law summarises a curve; it is not an oracle for each point.

WORKED EXAMPLE: THE LENGTH TRAP IN ACTION

Six genres of the Brown corpus. Raw TTR uses the whole genre. Standardised TTR averages over non-overlapping 8,000-token windows.

Genre	N	raw TTR	rank	std. TTR	rank
press reportage	90,089	0.1359	3	**0.3077**	**1**
humour	18,450	**0.2552**	**1**	0.3021	2
general fiction	58,605	0.1458	2	0.2744	3
romance	58,966	0.1307	4	0.2520	4
learned	164,036	0.0885	6	0.2478	5
government	63,214	0.1106	5	0.2320	6

- Raw TTR crowns humour, which is also the shortest genre at 18,450 tokens. The measurement is reading length
- Raw TTR ranks fiction above press reportage. On equal windows the order reverses
- Learned prose looks worst on raw TTR and is mid-table on equal windows. It is long, not poor

- **Standardised TTR (MSTTR).** Cut the text into equal windows of fixed size W, compute TTR in each, and average:

$$\mathrm{MSTTR} = \frac{1}{m} \sum_{j=1}^{m} \frac{V_j}{W}$$

  Simple, and the one used in the table above. Its weakness is that the answer depends on W
- **Root TTR.** $V/\sqrt{N}$, which is Heaps' law with b forced to 0.5. It only removes the length effect if b really is 0.5
- **Report (k, b) instead.** The honest summary of lexical richness is the fitted Heaps curve, not a single ratio. Quote the pair, because k and b are coupled

---

Never compare type–token ratios across samples of different sizes.

APPLICATIONS OF TTR

- Monitor vocabulary usage in a text or by a writer
- Track child vocabulary development, where sample size is controlled by design
- Estimate vocabulary variation across genres or authors
- Detect templated or repetitive machine-generated text. A decoder stuck in a loop shows a collapsing TTR
- Diagnose degenerate decoding. Low-temperature and greedy generation both drive TTR down, which is a symptom you will meet again in session 19

EXERCISES

1. Fit α on your own corpus by least squares over ranks $1 \ldots 100$, then over all ranks. Report both, with the ranges. Explain the difference in one sentence.
2. Refit with Mandelbrot's β and report (α, β) as a pair. Show that a range of pairs fits almost equally well.
3. Compute the coverage of the top 10, 100 and 1000 words directly from your counts. Compare with the harmonic-number prediction using your fitted α.
4. Fit Heaps' law on the first tenth of your corpus, predict V for the whole, then measure it. Report the signed error.
5. Find the percentage of tokens that are stop words, and the percentage of *types* that are stop words. Explain why the two numbers are so far apart.
6. Compare the TTR of two genres, first raw and then on equal windows. Report whether the ranking changes.

Predict every number before you compute it. The gap between prediction and measurement is where the marks are.

OPEN-CLASS AND STOP WORDS

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

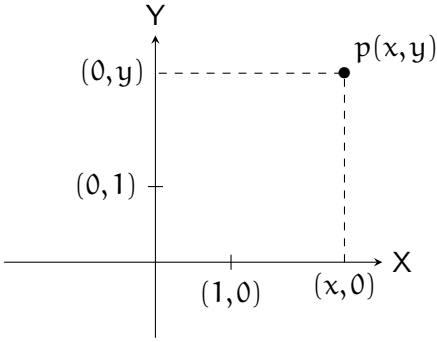
- Contribute to the overall meaning of a sentence
- Include nouns, verbs, adjectives, and adverbs
- **Expansive**: New words can be added to this category over time (e.g., "vlog," "emoji", “Mulligatawny”)
- **Adaptability**: New terms to express emerging concepts, technologies, and innovation

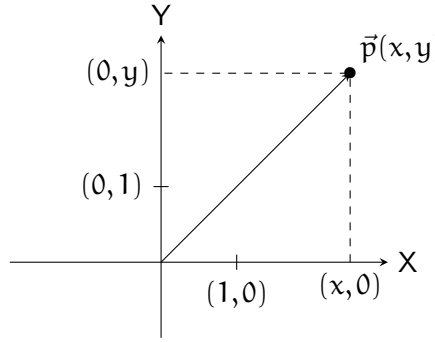
Closed-class (function) words, the usual stop words, serve grammatical purposes

- 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

2-D VECTOR SPACE

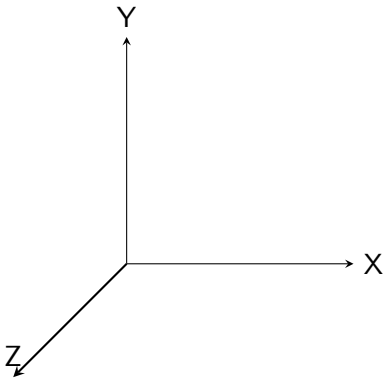
A 2-D vector-space is defined as a set of linearly independent basis vectors with 2 axes. Each axis corresponds to a dimension in the vector-space

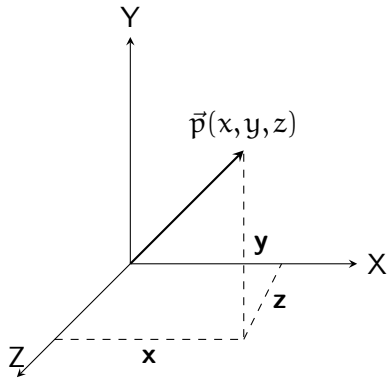




3-D VECTOR SPACE

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

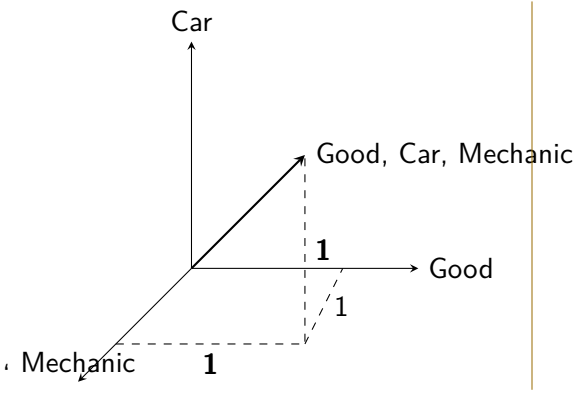
- Vector space models are used to represent words in a continuous vector space $\mathscr{R}$
- Combination of Terms represent a document vector in the word vector space

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

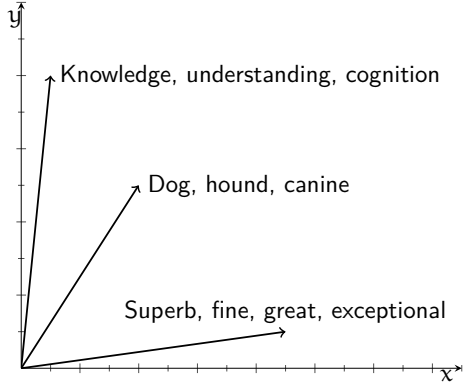
$$\left(t^{\text{House}}\right)^{T} \cdot t^{\text{Apartment}} = 0 \tag{19}$$

$$\left(t^{\text{Home}}\right)^{T} \cdot t^{\text{House}} = 0 \tag{20}$$

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

RELATIONSHIP AMONG TERMS - SYNONYMS

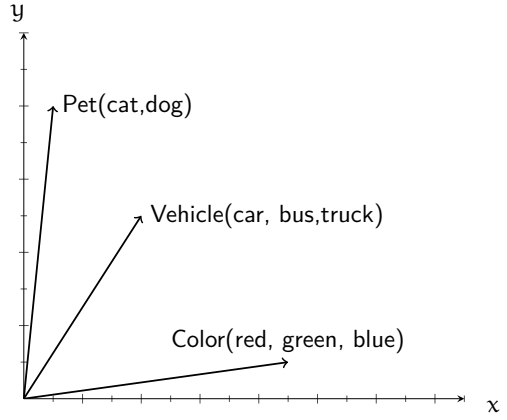
We could represent all the synonyms of a word in one axis



- 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*)అద్భుతమైన
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.

SIMILARITY MEASURES

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.

$$\textbf{Euclidean Distance} - \mathbb{E}(\vec{w_1}, \vec{w_2}) = \sqrt{w_1^2 - w_2^2} \tag{21}$$

$$\underset{\in[-1,1]}{\textbf{Cosine Similarity}} = \frac{\vec{w_1}.\vec{w_2}}{\|\vec{w_1}\| \, \|\vec{w_2}\|} \tag{22}$$

$$\underset{\in[0,1]}{\textbf{Cosine Distance}} = 1 - \textbf{Cosine Similarity} \tag{23}$$

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)

REFERENCES I

[1] George Kingsley Zipf. "The psychology of language". In: *Encyclopedia of psychology*. Philosophical Library, 1946, pp. 332–341.