Workshop on Natural Language Processing (Day 1)

Ramaseshan Ramachandran

Introduction

What's new?

How's it going?

How have you been?

How's everything?

How are you holding up?

What's up?

How are you?

What are you up to?

How are things going?

What's happening?

What's going on?

How's life?

How are you doing?

NATURAL LANGUAGE

- Allows interaction among humans to share information using a set of words and sentences constructed using a finite set of alphabets and framed using a set of grammar rules

Arbitrary
Structured
Generative
Dynamic

PROGRAMMING LANGUAGE

Intended for Human Machine Communications

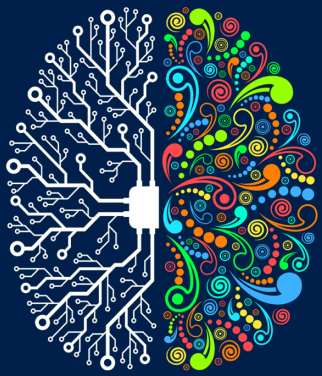
- Instructions are
  - Precise
  - Unambiguous
  - Mathematical equations

IS NLP HARD?

- Local high school dropouts cut in half
- The king saw a rabbit with his glasses
- Juvenile court to try shooting defendant
- What is added with 15 to get 45?
- Safety experts say school bus passengers should be belted

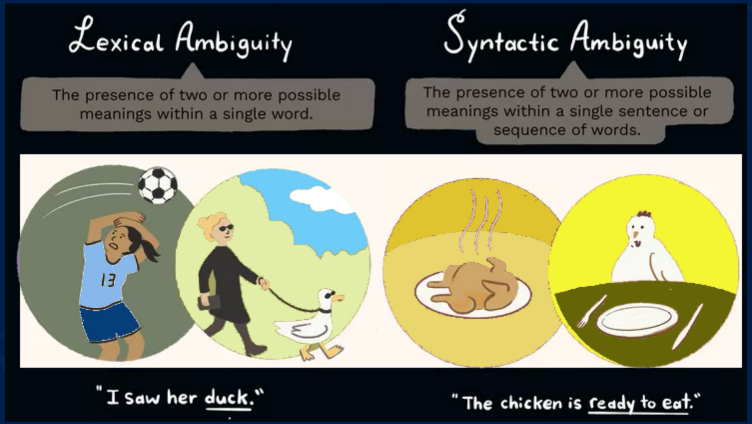
WHY NLP IS HARD? I

Creative and Analytical Represention of Ideas



WHY IS NLP HARD? II

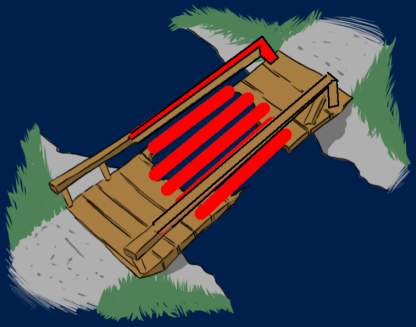
Ambiguous in nature



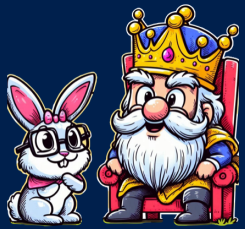
Source: Definition and Examples of Ambiguity in English(thoughtco.com)

Ambiguous in nature

Red tape holds up new bridges



PRAGMATIC CONSIDERATIONS



WHY IS NLP HARD? IV

Multiple representations of the same scenario

- The weather is extremely cold today
- It is freezing out there

SRACASM IS HARD EVEN FOR HUMANS

Irony, sarcasm (Yeah! right), double negatives, etc. make it difficult for automatic processing

Me: I am 25 years old
Mirror: Yeah, right

COMPLEX REPRESENTATION

Complex representation information (simple to hard vocabulary with uncommon usage of a sentence)

He was, in the way of most men, possessed of a rudimentary intelligence, his countenance ordinary, his bearing mild, with some weakness about the shoulders, his hair the color of ash; he spoke of the weather

The complex houses married and single soldiers and their families

TYPICAL TASKS

Information Retrieval	Find documents based on keywords
Information Extraction	Identify and extract personal name, date, company name, city..
Language generation	Description based on a photograph Title for a photograph
Text clustering	Automatic grouping of documents
Text classification	Assigning predefined categorization to documents Identify Spam emails and move them to a Spam folder
Machine Translation	Translate any language Text to another (when one language is unknown)
Grammar checkers	Check the grammar for any language

- Sentiment Analysis Search Engines
- Content or News curation
- Automatic Machine Translation Chatbots
- Transcription of Text from Audio/Video

Ability to harness information from a large corpus of text with no human intervention

IDEAL PROPERTIES OF A CORPUS

- Corpus is huge – Several billions of words
- Useful to verify a hypothesis about a language
- Find usage of a particular sound, word, or syntactic construction varies in different contexts
- Collection of most of the words of a language
- Even distribution of texts from all domains of language use

- The boys play cricket on the river bank.
- The boys play cricket by the side of a nationalized bank

Corpus Capital

LEXICAL RESOURCES

- Brown Corpus contains a collection of written American English
- Sussane is a subset of Brown, but is freely available
- A bi-lingual parallel corpus, Canadian Hansards, contains French and English transcripts of the parliament
- Penn-Treebank contains annotated text from the Wall Street journal
- Most NLP software platforms such as NLTK, Spacy include several corpora for learning purposes
- HuggingFace and Kaggle - Several corpora text and image for machine learning applications
- Wiki dumps for various languages

IMPORTANT ASPECTS OF NLP APPLICATION

- Dependent
- Consistent
- Transparent

To achieve Human like performance, we require all the three aspects well defined and implemented

We cannot solely dependent on statistical analysis or machines learning

They are useful in making a good guess based on the historical information

We require theoretical insights to provide human-like performance

COMMON LAYERS OF NLP APPLICATIONS

- Preprocessing layer
- Data extraction layer
- Analysis of extracted information
- Semantic understanding
- Human/automatic evaluation of word meaning, sentence structure using the content obtained from the previous layers

OPERATIONS ON A CORPUS

- Text normalization - Converting text into a single canonical form - removal of foreign words, case folding, ...
- Tokenization - This is the process of dividing input text into tokens/words by identifying word boundary
- Identification/Extraction Process of identifying certain tokens, sentences and paragraphs
- The number of tokens/words in a corpus
- Vocabulary count

Text Analytics

The set of unique words used in a corpus is referred to as the vocabulary

$$V = t_1, t_2, t_3, \cdots, t_n$$

- Describe word–rank and word–frequency distribution, vocabulary, and terms in a corpus

Empirical Laws

Heaps	Zipf	Mandelbrot
• Relates the terms and the vocabulary	• Relates frequency of a word and its rank	• A better approximation of Zipfs Law

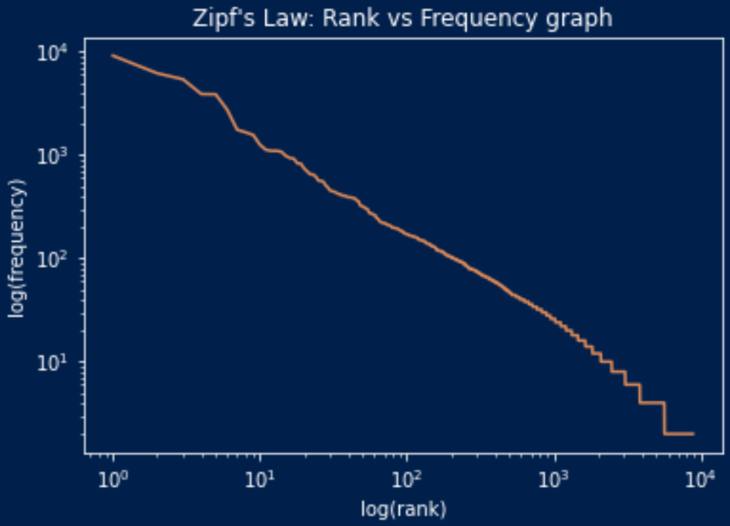
ZIPF'S LAW

The frequency of any word is inversely proportional to its rank

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

where $\alpha \approx 1$, r is the frequency rank of a word and f is the frequency of the word in the corpus.

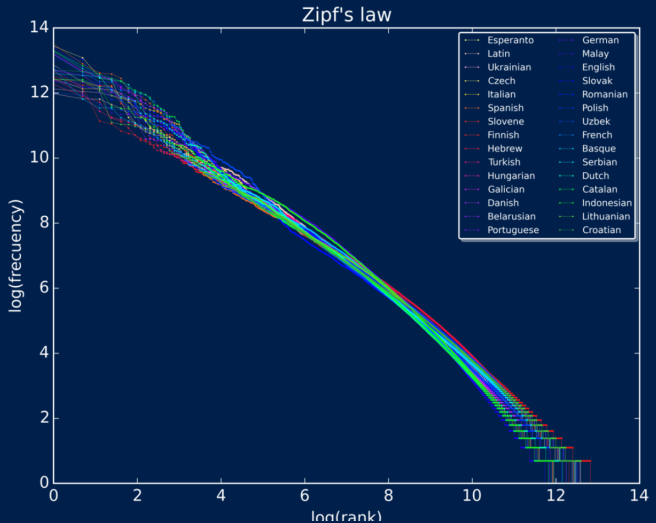
ZIPF'S LAW



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

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

ZIPF'S LAW



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

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.

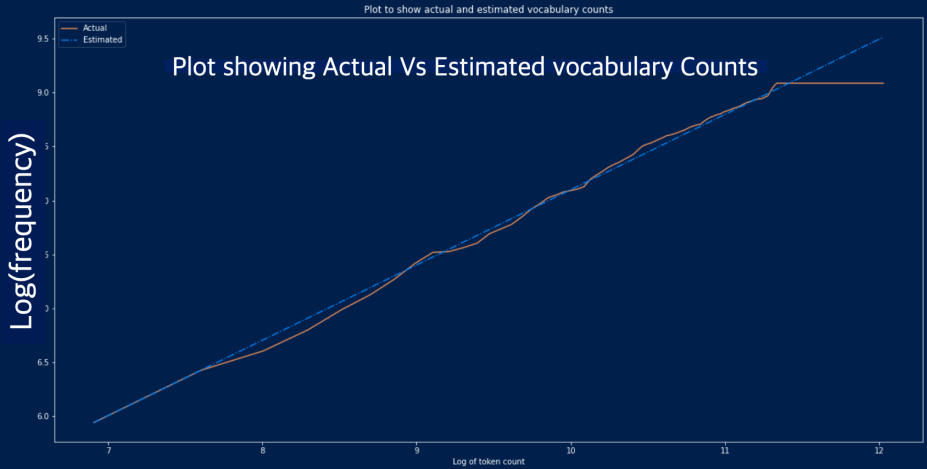
Source: Zipf's law -Wikipedia

MANDELBROT LAW

The frequency of any word is inversely proportional to its rank. Mandelbrot derived a more generalized law to closely fit the frequency distribution in language by adding an offset to the rank

$$f \propto \frac{1}{(r+\beta)^{\alpha}}, \text{ where } \alpha \approx 1 \text{ and } \beta \approx 2.7$$

HEAP'S LAW


Plot to show actual and estimated vocabulary counts
Actual
Estimated
Plot showing Actual Vs Estimated vocabulary Counts
Log(frequency)
9.5
9.0
6.5
6.0
7
8
9
10
11
12
Log of token count
Log(token count)

TERM FREQUENCY

Term Frequency

Term frequency is defined as the number of times the term, t_i, occurs in a document d_j, belonging to a corpus $(d_1, d_2, d_3, \ldots, d_m)$. This is denoted by $tf_{t,d}$

TERM FREQUENCY - DEMO

```
import nltk
from nltk.probability import  FreqDist
from nltk.corpus import stopwords
import pandas as pd

def process_text(file_id):
    stop_words = set(stopwords.words('english'))
    words = nltk.Text(nltk.corpus.gutenberg.words(file_id))
    words = [word.lower() for word in words if word.isalpha() and word.lower() not in stop_words]
    return words

def find_word_frequency(words, top_n=10):
    freq_dist = FreqDist(words)
    heading = ['Word', 'Frequency']
    tf_list = [(x, v) for x, v in freq_dist.most_common(top_n)]
    return pd.DataFrame(tf_list, columns=heading)

def find_weighted_word_frequency(words, top_n=10):
    freq_dist = FreqDist(words)
    heading = ['Word', 'Weighted Frequency']
    tf_list = [(x, v / len(freq_dist)) for x, v in freq_dist.most_common(top_n)]
    return pd.DataFrame(tf_list, columns=heading)

if __name__ == '__main__':
    words = process_text('bryant-stories.txt')
    print(find_word_frequency(words))
    print(find_weighted_word_frequency(words))
```

RAW AND ADJUSTED TERM FREQUENCY

word	Raw Frequency	Adjusted Frequency*
little	597	0.162
said	453	0.126
came	191	0.052
one	183	0.050
could	158	0.043
king	141	0.038

*The term frequency is adjusted to the document length

Demo

Representation of Words

WORD AS ATOMIC UNIT

- How do we represent the words?
- How do you present it as input to the machine?
- Can the word be used as the atomic unit?

CONTEXTUAL DECODING

The view from the mountain was
The scenery from the mountain was
The view from the summit was
The outlook from the peak was
The landscape seen from the mountain was

Awesome
Breathtaking
Amazing
Astonishing
Incredible
അത്ഭുതകരമായ
शानदार
চমকপ্রদ/আশ্চর্যজনক
kupanaha (Hawaiian)
అద్భుతమైన
Großartig
ರೀಕಿ ಐ°
伟大的/Wěidà de
素晴らしい

How do we represent the word?

Real – 1.31

Complex – 13.1+17.2i

One-Hot Vector – |0 0 0 0 1|

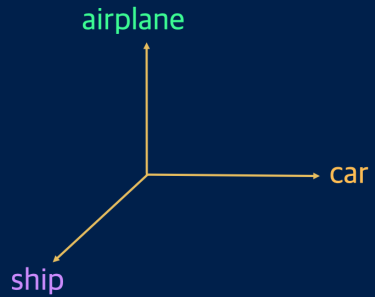
Sparse Vector – |0.2 0 0 0 1.3|

DenseVector – |-1.43 2.18 0.02 -0.84 0.123|

DISADVANTAGES OF USING TERM FREQUENCY TO REPRESENT A WORD

- All terms are given equal importance
- The Common term the has no relevance to the document, but gets high relevancy score
- Repetition of frequency values possible
- Unique representation not possible

Let us assume that the vocabulary of a corpus are considered as linearly independent basis vectors.



The vocabulary size of Google News Corpus is 3 million words. If we plot all the words in the real space R, we get 3 million axes

A similarity measure is a real-valued function that quantifies the similarity between two vectors. Most used one is Cosine Similarity

$$C_s = \frac{\overrightarrow{w_1} \cdot \overrightarrow{w_2}}{||w_1||||w_2||}$$

$$C_d = 1 - \frac{\overrightarrow{w_1} \cdot \overrightarrow{w_2}}{||w_1||||w_2||}$$

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 beween two words (w_x, w_y) is a function of how frequently they appeared 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 the similar words (synonyms, hyponyms, hypernyms) are classified and arranged in various axes
  - Colour (hypernym) - $\underbrace{\text{red}, \text{Green}, \text{Orange}}_{\text{co-hyponyms}}$ (hyponym) - Attributional Similarity
- A space where the similar words (synonyms, hyponyms, hypernyms) are classified and arranged in various axes
- A model or models that automatically find similar words are known as Distributed 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 cook, cooked, cooks and cooking are related due to its co-occurrence statistics - semantic relationship
- The words dinner and Sunday are similar due to associative relationship and due to 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 pain, illness, blood, drugs, syringe, but not to thread, knitting, cloth, etc.

Associative relationship

You shall know a word by the company it keeps

- Firth, 1957

WORDS AND TERMS

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

- The atomic unit for constructing a word in a language is its alphabet
- We use ***Term*** (co-located/co-occurring words) and ***word*** as atomic.
- It is necessary to consider the numerical representation of the word for computational purposes
- Vocabulary of size $N = 1 \ldots n$ defined as $V = w_1, w_2, w_3, \ldots, w_n$ is the vocabulary containing unique words of a corpus
- Some words found in V appear in documents ($D = D_1, D_2 D_3, \ldots, D_m$), once or several times or may not appear at all.

MULTIPLE WEIGHTING FACTORS TF

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

$$\mathrm{RawCount} - \mathrm{tf}_{i,d} \tag{2}$$

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

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

DISADVANTAGES OF RAW FREQUENCY

- All terms are given equal importance
- The Common term ***the*** has no relevance to the document, but gets high relevancy score
- May not be suitable for classification when common words appear in documents

INVERSE DOCUMENT FREQUENCY

In order to attenuate the effect of frequently occurring terms, it is important to scale it down and at the same time it is necessary to increase the weight of terms that occur rarely.
Inverse document frequency (IDF) is defined as

$$\mathrm{IDF_t} = \log\left(\frac{\mathrm{N}}{\mathrm{D_{f_t}}}\right) \tag{5}$$

where N is the total number of documents in a collection, and $\mathrm{D_{f_t}}$ is the count of documents containing the term t

- Rare documents gets a significantly higher value
- Commonly occurring terms are attenuated
- It is a measure of informativeness
- Reduce the tf weight of a term by a factor that grows with its collection frequency.
- If a term appears in all the documents, then IDF is zero. This implies that the term is not important

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 is the inverse frequency of the word 't' appearing in the corpus. 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 words in the corpus = 100000

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

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

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 the TF-IDF, the rank order for the documents can be determined for the documents for the 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

The collection $[tf_1, tf_2, tf_5, tf_{15}, \ldots, \ldots, tf_n]$ is known as *bag of words*

- The ordering of the terms is not important
- Two documents with similar bag of words are similar in content
- It refers to the quantitative representation of the document

OPEN-CLASS AND STOP WORDS

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

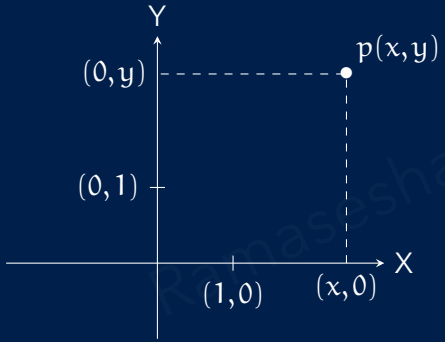
- Contribute to the overall meaning of a sentence
- Include nouns, verbs, adjectives, and adverbs
- Expansive: New words can be added to this category over time (e.g., "vlog," "emoji", "Mulligatawny")
- **Adaptability**: New terms to express emerging concepts, technologies, and innovation

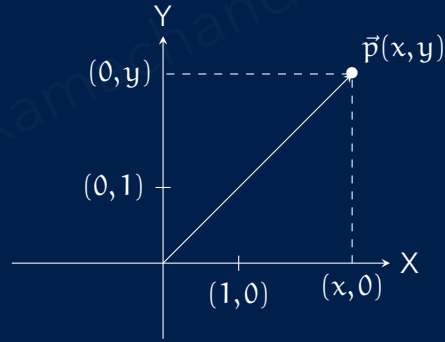
Stop words: Serve grammatical purposes

- 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

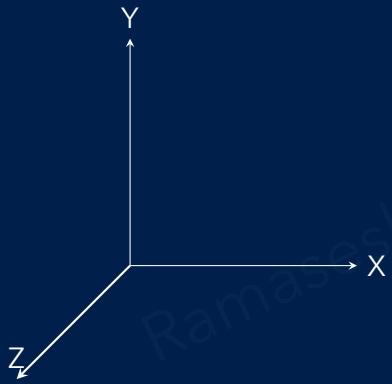
A 2-D vector-space is defined as a set of linearly independent basis vectors with 2 axes. Each axis corresponds to a dimension in the vector-space

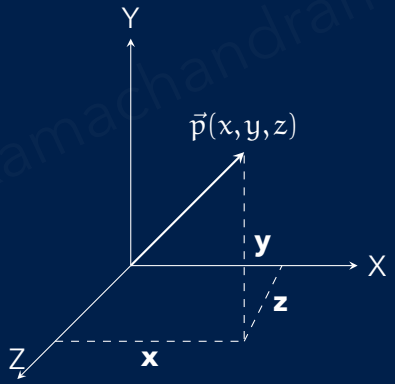




3-D VECTOR SPACE

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.

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.

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

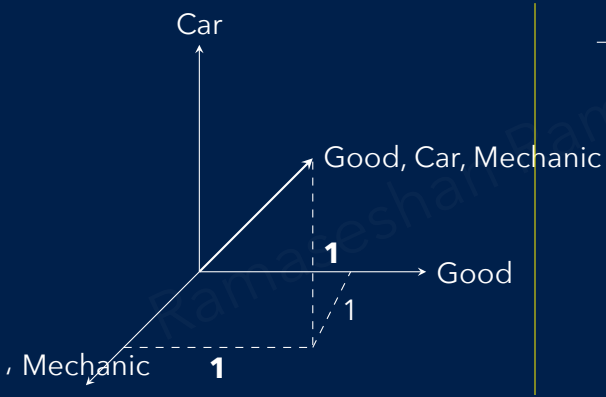
- Vector space models are used to represent words in a continuous vector space $\mathscr{R}$
- Combination of Terms represent a document vector in the word vector space

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?

WORD SIMILARITY

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.

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

$$\text{Cosine Similarity} = \frac{\vec{w_1}.\vec{w_2}}{\|\vec{w_1}\| \, \|\vec{w_2}\|} = \frac{\vec{w_1}}{\|\vec{w_1}\|} . \frac{\vec{w_2}}{\|\vec{w_2}\|}$$

$$\text{Cosine distance} = 1 - \frac{\vec{w_1}.\vec{w_2}}{\|\vec{w_1}\| \, \|\vec{w_2}\|} = \frac{\vec{w_1}}{\|\vec{w_1}\|} . \frac{\vec{w_2}}{\|\vec{w_2}\|}$$

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

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

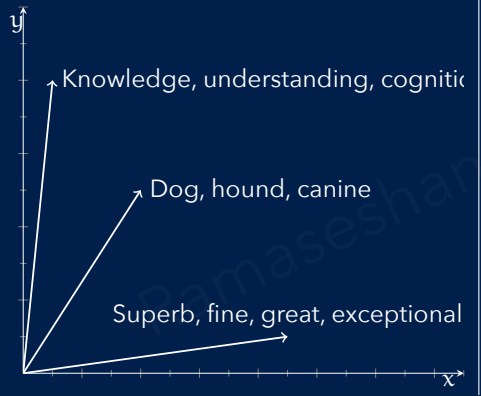
$$\left(t^{\text{House}}\right)^{\text{T}} \cdot t^{\text{Apartment}} = 0 \tag{7}$$

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

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

RELATIONSHIP AMONG TERMS - SYNONYMS

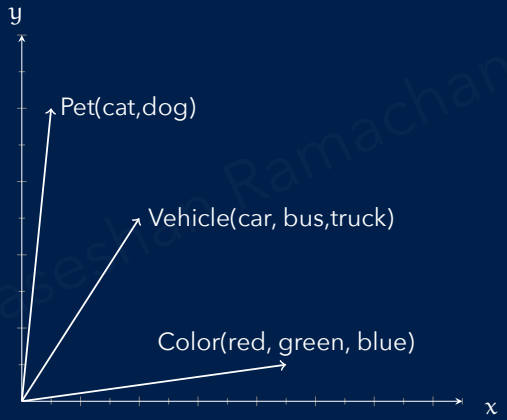
We could represent all the synonyms of a word in one axis



- 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/चमकप्रद
astonishing
awe-inspiring
extraordinary
incredible/(*incroyable*)
unbelievable
magnificent शानदार/ಗಂಭೀರಮಾಯ/ಅಪ
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

Intuition

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

Aji Amarillo is a member of pepper family

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
- 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

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.

$$\text{Euclidean Distance - } \mathbb{E}(\vec{w_1}, \vec{w_2}) = \sqrt{w_1^2 - w_2^2} \tag{9}$$

$$\text{Cosine Similarity} = \frac{\vec{w_1}.\vec{w_2}}{\|\vec{w_1}\| \, \|\vec{w_2}\|} \tag{10}$$

$$\text{Cosine distance} = 1 - \textbf{Cosine Similarity} \tag{11}$$

WORD VECTOR EXAMPLES

```
Similar words for apple
('apple',    0)
('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 - american

```
'american', 0
'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

$$\overrightarrow{\text{apple}} - \overrightarrow{\text{iphone}}$$

```
('raisin', 0.5744591153088133)
('pecan', 0.5760617374141159)
('cranberry', 0.5840016172254104)
('butternut', 0.5882322018694753)
('cider', 0.5910795032086132)
('apricot', 0.6036644437522422)
('tomato', 0.6073715970323961)
('rosemary', 0.6150986936477657)
('rhubarb', 0.6157884153793192)
('feta', 0.6183016129045151)
('apples', 0.6226003361980218)
('avocado', 0.6235366677962004)
('fennel', 0.6306016018912576)
('chutney', 0.6312524337590703)
('spiced', 0.6327632200841328)
```

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)

Hyperspace Analogue To Language[1] (HAL)

[1] K. Lund and C. Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. Be- havior Research Methods, Instruments, & Computers, 28(2):203–208, 1996.

HIGHLIGHTS OF THE HAL PAPER

Motivation

Human semantic memories are presumably constructed through experience with the world; as concepts are encountered, information about their meanings is accumulated.

Two problems of constructing semantic spaces manually

- Find a set of axes that define a concept
- Determine where each word should fall on each axis

This is a tedious process and an error-prone

HAL METHODOLOGY

1. A **weighted window** representing a span of words(n-grams) is moved across the corpus in one-word increments.

   Example

   Small corpus: A weighted window representing a span of words(n-grams) is moved across the corpus in one-word increments.

   n-grams

   ('By', 'moving', 'this') ('moving', 'this', 'window') ('this', 'window', 'over') ('window', 'over', 'the') ('over', 'the', 'source') ('the', 'source', 'corpus') (⋯)

2. Capture the co-occurrence values of the words within it at every window movement to form a co-occurrence matrix.
3. Each cell of the matrix represents the summed co-occurrence counts for a single word pair (w_t, w_n)
4. The accumulation of values is direction sensitive
   The count of sequence "$w_1 w_2$" and count for the sequence "$w_2 w_1$" are different
5. For every word, there is both a row and a column containing relevant co-occurrence values, each one representing its concept axis

HAL SCANNING

Corpus: the horse raced past the barn fell

Left2Right Scanning

the	horse	raced	past	the	barn	fell
K	5	4	3	2	1	0

	horse	raced	past	the	barn	fell
	K	5	4	3	2	1

Right2Left Scanning

fell	barn	the	past	raced	horse	the
K	5	4	3	2	1	0

	barn	the	past	raced	horse	the
	K	5	4	3	2	1

Incidence Matrix

	the	horse	raced	past	barn	fell
the	2	3				
horse	5					
raced	4	5				
past	3	4				
barn	1	2				
fell		1				

	the	horse	raced	past	barn	fell
the						
horse						
raced						
past						
barn						
fell	4	1	2	3	5	

HAL ALGORITHM

Require a big corpus >5 GB for a reasonable similarity measures

1. Preprocess to limit the vocabulary size
2. Perform two scans using a ramping window of size 11 - first $\rightarrow$ direction and later in the $\leftarrow$ direction
3. Use the first word as the key word and the rest as context words
4. Use the last word as the key and rest as the context words, during the $\leftarrow$ scanning
5. The nearest neighbor of the key gets the weight 10 and the 10th word gets the weight 1
6. Construct an incidence matrix using the co-occurrence values
7. Concatenate two word vectors found for every word (row and column) in the matrix Concatenate them to get the word vectors for all the words in the vocabulary.
8. The number of elements in the word vector will be $2||V|$

HAL EXPERIMENT

160 million words from Usenet news groups

Window size = 10

- Vocabulary - Words with a frequency > 50
- Zipf's law is used to eliminate most common and rare words
- Minkowski distance measure is used for computing word similarities

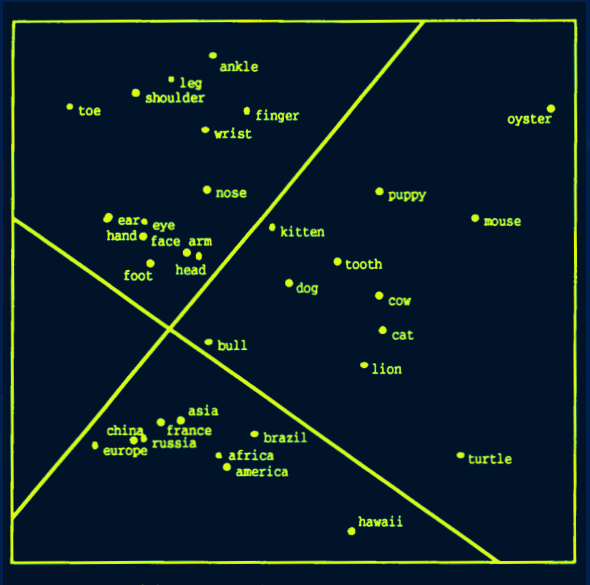
$$d_{x_i y_j} = \sqrt[r]{|x_i - y_i|^r}$$

- The word vectors produce high dimensional semantic space - associative
- This is an unsupervised analysis of text
- Demonstrated sizable correlation between vector similarity and basic cognitive effects

RESULTS

Target	n1	n2	n3	n4	n5
jugs	juice	butter	vinegar	bottles	cans
leningrad	rome	iran	dresdan	azerbaijan	tibet
lipstick	lace	pink	cream	purple	soft
triumph	beauty	prime	grand	former	rolling
cardboard	plastic	rubber	glass	thin	tiny
monopoly	threat	huge	moral	gun	large

HAL WORD VECTORS - SIMILARITY CHART



CONCLUDING REMARKS - HAL

- HAL acquires contextual understanding of words by using the moving window and weighting co-occurrence distance.
- Using a large corpus, the co-occurrence matrix carries the history of this contextual experience
- The semantic vectors are representations that are essentially measures of context

IMPACT OF FREQUENCY MEASURE ON SIMILARITY

- Even if t_1 and t_2 are unrelated, if $p(t_1) \approx p(t_2)$, then their vectors will contain elements with similar magnitudes.

  $\implies$ any similarity measure

  For example, words **a, an, the** co-occur with many words in the vocabulary

- Conversely if they are related but $p(t_1) \ll p(t_2)$ then their vectors will contain elements with widely differing magnitudes, simply due to their differing co-occurrence probability.

In general, relative frequency does not imply semantic similarity. Hence we require normalized measures ro build word vectors.

HAL SUMMARY

- A count-based word embedding model
- Captures word meanings through the unsupervised analysis of text
- Produces word vectors that are semantic (similar words) and associative in nature
- Acquires word meanings as a function of keeping track of how words are used in context
- Carries the history of the contextual experience by using a moving window and weighting of co-occurring words based on the distance
- Exploits the regularities of language such that conceptual generalisations can be captured in a data matrix

IMPACT OF FREQUENCY MEASURE ON SIMILARITY

- Even if t_1 and t_2 are unrelated, if $p(t_1) \approx p(t_2)$, then their vectors will contain elements with similar magnitudes.

  $\implies$ any similarity measure

  For example, words **a, an, the** co-occur with many words in the vocabulary

- Conversely if they are related but $p(t_1) \ll p(t_2)$ then their vectors will contain elements with widely differing magnitudes, simply due to their differing co-occurrence probability.

In general, relative frequency does not imply semantic similarity. Hence we require normalized measures ro build word vectors.

HAL Demo

Correlated Occurrence Analogue to Lexical Semantic[2] (COALS)

[2] D. L. Rohde, L. M. Gonnerman, and D. C. Plaut. An improved model of semantic similarity based on lexical co-occurrence. Communications of the ACM, 8(627-633):116, 2006.

COALS METHODOLOGY

- Gather co-occurrence counts, typically ignoring closed-class neighbors and using a ramped window of size 4.
- Discard all but the m (14,000, in this case) columns reflecting the most common open-class words.
- Convert counts to word pair correlations - Instead of using the raw frequency score, correlation score is used to analyze the relationship between pair of words
- The correlation coefficient values with this normalization will be in the range of [-1,1]
- Set negative values to 0.
- Take square root of positive values.
- The semantic similarity between two words is given by the correlation of their vectors.
- The matrix constructed using this correlation would be semantic space
- COALS method employs a normalization strategy that largely factors out lexical frequency.
- Columns representing low-frequency words are removed

NORMALIZATION PROCEDURES

Several vector normalization procedures.

Row:	$w'_{a,b} = \frac{w_{a,b}}{\sum_j w_{a,j}}$
Column:	$w'_{a,b} = \frac{w_{a,b}}{\sum_i w_{i,b}}$
Length:	$w'_{a,b} = \frac{w_{a,b}}{(\sum_j {w_{a,j}}^2)^{1/2}}$
Correlation:	$w'_{a,b} = \frac{T w_{a,b} - \sum_j w_{a,j} \sum_i w_{i,b}}{\left(\sum_j w_{a,j} \cdot (T - \sum_j w_{a,j}) \cdot \sum_i w_{i,b} \cdot (T - \sum_i w_{i,b})\right)^{1/2}}$ $T = \sum_i \sum_j w_{i,j}$
Entropy:	$w'_{a,b} = \log(w_{a,b} + 1)/H_a$ $H_a = -\sum_j \frac{w_{a,b}}{\sum_j w_{a,j}} \log\left(\frac{w_{a,b}}{\sum_j w_{a,j}}\right)$

$$\mathrm{PMI}(w_i, w_c) = \log_2\left(\frac{p(w_i, w_c)}{p(w_i)p(w_c)}\right)$$

The range of PMI is $[-\infty, \infty]$. Positive PMI refers to word that often co-occur, while negative indicates that they are almost independent of each other or they co-occur less often. To focus only on similarity, we can consider only positive PMI as follows:

$$\mathrm{PPMI}(w_i, w_c) = \max\left(\log\left(\frac{p(w_i, w_c)}{p(w_i)p(w_c)}\right), 0\right)$$

BIAS IN NORMALIZATION

All normalization procedures have bias. PMI score is high for rare words. One way to reduce this bias toward low frequency words is to scale down the context count for PMI as follows:

$$\text{PPMI}(w_i, w_c)_b = \max\left(\log\left(\frac{p(w_i, w_c)}{p(w_i)p_b(w_c)}\right), 0\right)$$

$$p_b(c) = \frac{\text{count}(c)^b}{\sum_c \text{count}(c)^b}$$

where $b \approx 0.75$

$b \approx 0.75$ increases the $p_b(c)$ Hence $p_b(c) > p(c)$

COALS ALGORITHM

1. Gather co-occurrence counts, typically ignoring closed-class neighbors and using a ramped, size 4 window:

   1 2 3 4 [0] 4 3 2 1

2. Discard all but the m (14,000, in this case) columns reflecting the most common open-class words.
3. Convert counts to word pair correlations, set negative values to 0, and take square roots of positive ones.
4. The semantic similarity between two words is given by the correlation of their vectors.

BUILDING CO-OCCURRENCE MATRIX - COALS STEP 1

How much wood would a woodchuck chuck, if a woodchuck could chuck wood?
As much wood as a woodchuck would,if a woodchuck could chuck wood.

Step 1 of the COALS method: The initial co-occurrence table with a ramped, 4-word window.

	a	as	chuck	could	how	if	much	wood	woodch.	would	,	.	?
a	0	5	9	6	1	10	4	8	18	9	10	0	0
as	5	4	2	1	0	0	7	10	3	2	1	0	5
chuck	9	2	0	8	0	5	1	9	11	2	4	3	3
could	6	1	8	0	0	4	0	6	8	0	2	2	2
how	1	0	0	0	0	0	4	3	0	2	0	0	0
if	10	0	5	4	0	0	0	0	10	3	8	0	0
much	4	7	1	0	4	0	0	10	2	3	0	0	3
wood	8	10	9	6	3	0	10	2	8	5	0	4	6
woodch.	18	3	11	8	0	10	2	8	0	8	10	1	1
would	9	2	2	0	2	3	3	5	8	0	5	0	0
,	10	1	4	2	0	8	0	0	10	5	0	0	0
.	0	0	3	2	0	0	0	4	1	0	0	0	0
?	0	5	3	2	0	0	3	6	1	0	0	0	0

COALS-STEP 2

Step 2 of the COALS method: Raw counts are converted to correlations.

	a	as	chuck	could	how	if	much	wood	woodch.	would	,	.	?
a	-0.167	-0.014	0.014	0.009	-0.017	0.085	-0.018	-0.033	0.096	0.069	0.085	-0.055	-0.079
as	-0.014	0.031	-0.048	-0.049	-0.037	-0.077	0.133	0.103	-0.054	-0.021	-0.050	-0.037	0.133
chuck	0.014	-0.048	-0.113	0.094	-0.045	0.021	-0.061	0.031	0.048	-0.046	-0.002	0.088	0.031
could	0.009	-0.049	0.094	-0.075	-0.037	0.033	-0.070	0.022	0.049	-0.075	-0.021	0.069	0.023
how	-0.017	-0.037	-0.045	-0.037	-0.018	-0.037	0.192	0.070	-0.055	0.069	-0.037	-0.018	-0.026
if	0.085	-0.077	0.021	0.033	-0.037	-0.077	-0.071	-0.106	0.085	0.006	0.138	-0.037	-0.053
much	-0.018	0.133	-0.061	-0.070	0.192	-0.071	-0.065	0.128	-0.061	0.019	-0.071	-0.034	0.072
wood	-0.033	0.103	0.031	0.022	0.070	-0.106	0.128	-0.113	-0.033	0.001	-0.106	0.111	0.100
woodch.	0.096	-0.054	0.048	0.049	-0.055	0.085	-0.061	-0.033	-0.167	0.049	0.085	-0.017	-0.051
would	0.069	-0.021	-0.046	-0.075	0.069	0.006	0.019	0.001	0.049	-0.075	0.060	-0.037	-0.053
,	0.085	-0.050	-0.002	-0.021	-0.037	0.138	-0.071	-0.106	0.085	0.060	-0.077	-0.037	-0.053
.	-0.055	-0.037	0.088	0.069	-0.018	-0.037	-0.034	0.111	-0.017	-0.037	-0.037	-0.018	-0.026
?	-0.079	0.133	0.031	0.023	-0.026	-0.053	0.072	0.100	-0.051	-0.053	-0.053	-0.026	-0.037

$$r = \frac{T w_{a,b} - \sum_j w_{a,j} \sum_i w_{b,i}}{\sqrt{\sum_j w_{a,j} (T - \sum_j w_{a,j}) \sum_i w_{b,i} (T - \sum_i w_{b,i})}}$$

$$\text{where } T = \sum_i \sum_j w_{i,j}$$

COALS - STEP 3

Step 3 of the COALS method: Negative values discarded and the positive values square rooted.

	a	as	chuck	could	how	if	much	wood	woodch.	would	,	.	?
a	0	0	0.120	0.093	0	0.291	0	0	0.310	0.262	0.291	0	0
as	0	0.175	0	0	0	0	0.364	0.320	0	0	0	0	0.365
chuck	0.120	0	0	0.306	0	0.146	0	0.177	0.220	0	0	0.297	0.175
could	0.093	0	0.306	0	0	0.182	0	0.149	0.221	0	0	0.263	0.151
how	0	0	0	0	0	0	0.438	0.265	0	0.263	0	0	0
if	0.291	0	0.146	0.182	0	0	0	0	0.291	0.076	0.372	0	0
much	0	0.364	0	0	0.438	0	0	0.358	0	0.136	0	0	0.268
wood	0	0.320	0.177	0.149	0.265	0	0.358	0	0	0.034	0	0.333	0.317
woodch.	0.310	0	0.220	0.221	0	0.291	0	0	0	0.221	0.291	0	0
would	0.262	0	0	0	0.263	0.076	0.136	0.034	0.221	0	0.246	0	0
,	0.291	0	0	0	0	0.372	0	0	0.291	0.246	0	0	0
.	0	0	0.297	0.263	0	0	0	0.333	0	0	0	0	0
?	0	0.365	0.175	0.151	0	0	0.268	0.317	0	0	0	0	0

COALS RESULTS

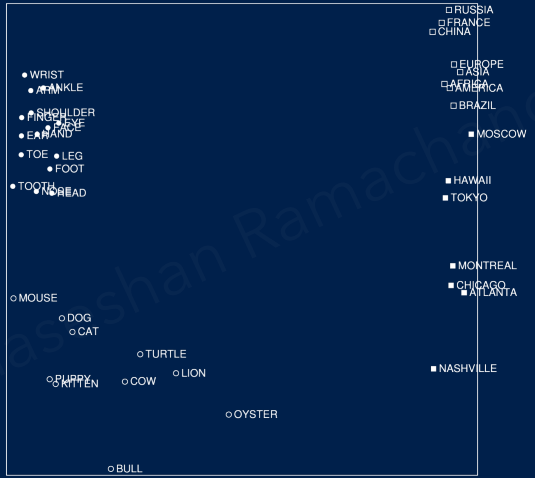
Nearest neighbors and their percent correlation similarities for a set of nouns

	gun	point	mind	monopoly
1)	46.4 handgun	32.4 points	33.5 minds	39.9 monopolies
2)	41.1 firearms	29.2 argument	24.9 consciousness	27.8 monopolistic
3)	41.0 firearm	25.4 question	23.2 thoughts	26.5 corporations
4)	35.3 handguns	22.3 arguments	22.4 senses	25.0 government
5)	35.0 guns	21.5 idea	22.2 subconscious	23.2 ownership
6)	32.7 pistol	20.1 assertion	20.8 thinking	22.2 property
7)	26.3 weapon	19.5 premise	20.6 perception	22.2 capitalism
8)	24.4 rifles	19.3 moot	20.4 emotions	21.8 capitalist
9)	24.2 shotgun	18.9 distinction	20.1 brain	21.6 authority
10)	23.6 weapons	18.7 statement	19.9 psyche	21.3 subsidies

COALS RESULTS - VERBS

	need	buy	play	change
1)	50.4 want	53.5 buying	63.5 playing	56.9 changing
2)	50.2 needed	52.5 sell	55.5 played	55.3 changes
3)	42.1 needing	49.1 bought	47.6 plays	48.9 changed
4)	41.2 needs	41.8 purchase	37.2 players	32.2 adjust
5)	41.1 can	40.3 purchased	35.4 player	30.2 affect
6)	39.5 able	39.7 selling	33.8 game	29.5 modify
7)	36.3 try	38.2 sells	32.3 games	28.3 different
8)	35.4 should	36.3 buys	29.0 listen	27.1 alter
9)	35.3 do	34.0 sale	26.8 playable	25.6 shift
10)	34.7 necessary	31.5 cheap	25.0 beat	25.1 altering

MULTIDIMENSIONAL SCALING



SOME INSIGHTS 1/2

- The majority of the correlations are negative
- Words with negative correlations do not contribute well to finding similarity than the ones with positive correlation
- Closed-class words (147) convey syntactic information than semantic - could be removed from the correlation table punctuation marks, she, he, where, after, ...

SOME INSIGHTS 2/2

- Positive Correlation means that two words often appear together. In other words, their contexts are similar.
  - CMI and Prodigy have a strong correlation
- Zero correlation means the pair of words are statistically independent. Hence no influence
  - CMI and Engineering_Drawing/blueprint have no inherent or direct relationship in terms or context.
- Negative correlation indicates an inverse relationship. For every word pair in this set, the second word in each pair is less likely to appear, and vice versa.
  - When talking about one of the specializations of CMI as the first word in a pair, some words, such as art, literature, philosophy, and religion, are less likely to appear together (although they may appear a few times).

FIRST AND SECOND ORDER ASSOCIATIONS OF DSM

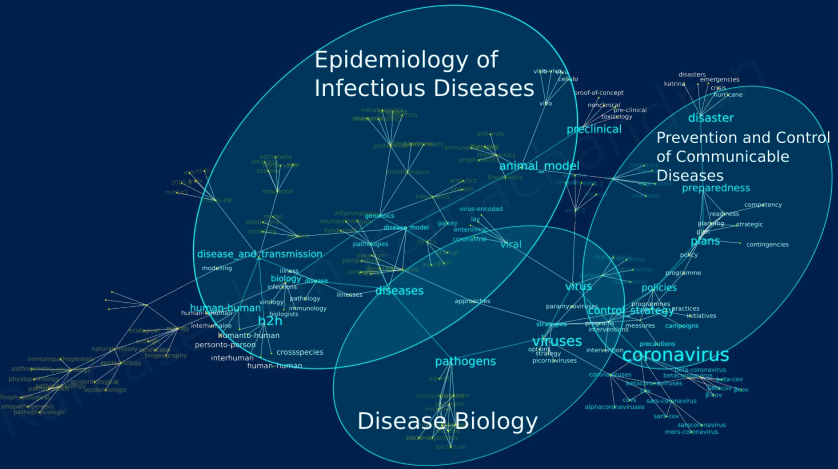
First Order Association

- Pairs of words in common contexts are semantically related
- If a word w_x occurred in several contexts along with w_y, then w_x and w_y are related by the first-order association. w_x and w_y are called as first-order associates.
- For any pair of words, w_i and w_j, if the strength of similarity is stronger, then they have a large number of common first-order associates

Second Order Association

- If a word w_y occurred in several contexts along with w_z in which w_x is absent (or occurred in a statistically insignificant number of times), then w_x and w_z are related by the second-order association. w_x and w_z are called as second-order associates.
  $\forall w_i, w_j, w_k \in V$, if we have $w_x R w_y$ and $w_y R w_z$, then $w_x R w_z$ where the relation R is transitive

MULTI-LEVEL WORD ASSOCIATIONS



GOAL

- Process each word in a Vocabulary of words to obtain a respective numeric representation of each word in the Vocabulary
- Reflect semantic similarities, Syntactic similarities, or both, between words they represent
- Map each of the plurality of words to a respective vector and output a single merged vector that is a combination of the respective vectors

Word2Vec

CONTEXT WORDS AND CENTRAL WORD

Continuous Bag of Words (CBOW) Model – A central word is surrounded by context words. Given the context words identify the central word

- Wish you many | more | happy | returns | of | the | day

Skip Gram Model – Given the central word, identify the surrounding words

- Wish you many | more | happy | returns | of | the | day

CONTINUOUS BAG OF WORDS (CBOW)

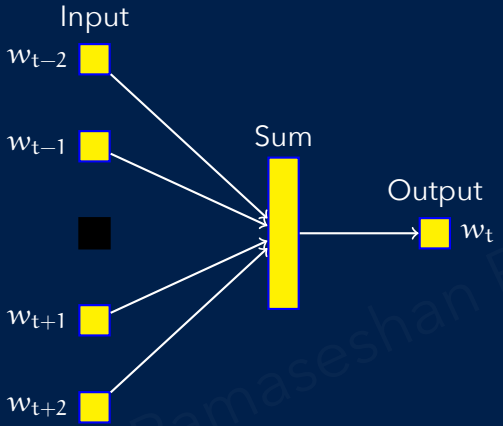


Figure: The CBOW architecture predicts the current word based on the context words of length n. Here the window size is 5

CBOW uses the sequence of words "Wish", "you", "a", "happy", "year" as a context and predicts or generates the central word "new"

- CBOW is used for learning the central word
- Maximize probability of word based on the word co-occurrences within a distance of n

SKIP GRAM MODEL

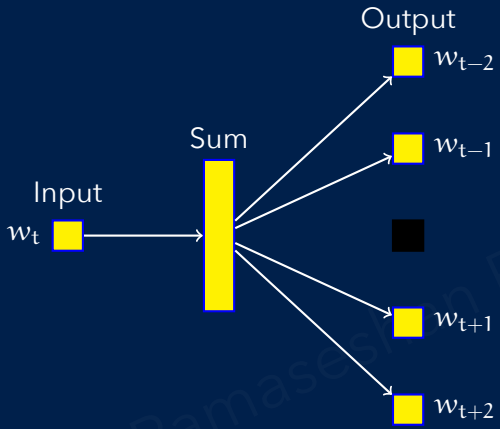


Figure: The SG architecture predicts the one context word at a time based on the center word. Here the window size is 5

SG uses the central word "new" and predicts the context words "Wish", "you", "a", "happy", "year"

- SG is used to learn the context words given the central word
- Maximize probability of word based on the word co-occurrences within a distance of $[-n, +n]$ from the center word

SOURCE PREPARATION FOR TRAINING

Source Text

Wish | you | many | more happy returns of the day→

Wish | you | more | happy | returns of the day→

Wish | you | many | more | happy | returns of the day→

Wish | you | many | more | happy | returns | of the day→

Wish you | many | more | happy | returns | of | the day→

Wish you many | more | happy | returns | of | the | day→

Wish you many more | happy | returns | of | the | day | →

Training Samples

$(wish, you)$
$(wish, many)$
$(you, Wish)$
$(you, more), (you, happy)$
$(many, Wish), (many, you)$
$(many, more), (many, happy)$
$(more, many), (more, you)$
$(more, happy), (more, returns)$
$(happy, many), (happy, more)$
$(happy, returns), (happy, of)$
$(returns, more), (returns, happy)$
$(returns, of), (returns, the)$
$(of, happy), (of, returns)$
$(of, the), (of, day)$

ONE-WORD LEARNING

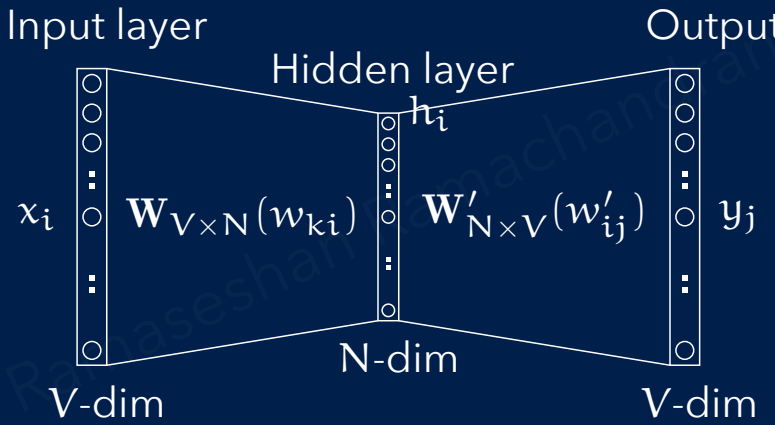


Figure: A CBOW model with only one word as input[1]. The layers are fully connected

Here the W and W' are learned

INPUT LAYER

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

$$x_k = 1 \text{ and } x'_k = 0, \forall k' \neq k$$

HIDDEN LAYER

This neural network is fully connected. Input to the network is a one-hot vector. v_w^T is the N-dimensional vector representation of the word presented as input[3]

$$\mathbf{h} = \mathbf{W^T X} \tag{12}$$

Now v_{wI} of the matrix (W) is the vector representation of the input one-hot vector w_I. From (12), h is a linear combination of input and weights.
In the same way. we get a score for $\mathrm{u_j}$

$$\mathrm{u_j} = \mathbf{v'^T_{w_j} h} = \mathbf{v'^T_{w_j} v_{w_I}} \tag{13}$$

where v_{w_I} is the vector representation of the input word w_I and v'_{w_j} is the $\mathrm{j^{th}}$ column of (W')

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space, 2013.

OUTPUT LAYER

At the output layer, we apply the softmax to get the posterior distribution of the word(s). It is obtained by,

$$p(w_j|w_I) = y_j \tag{14}$$

where y_j is the output of the j^{th} unit in the output layer

$$y_j = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u'_j)} \tag{15}$$

$$= \frac{\exp(\mathbf{v'^{T}_{w_j}} v_{wI})}{\sum_{j'=1}^{V} \exp((\mathbf{v'^{T}_{w_{j'}}}) v_{w_I})} \tag{16}$$

where $\mathbf{v}_w$, $\mathbf{v'}_w$ are the input vector (word vector) and output vector (feature vector) representations, of w_j and $w_{j'}$, respectively

UPDATE WEIGHTS - HIDDEN-OUTPUT LAYERS I

Through backpropagation,[4] the learning/training objective is to maximize the probability of the target word given the input word:

$$\max \mathrm{p}(w_{\mathrm{o}} \mid w_{\mathrm{I}}) = \max \log(\mathrm{y}_{\mathrm{j}^*}) \quad \text{(maximize log-likelihood)} \tag{17}$$

where $\mathrm{y}_{\mathrm{j}^*}$ is the probability assigned to the correct output word w_{o}, computed using the softmax function:

$$\mathrm{y}_{\mathrm{j}^*} = \frac{\exp(\mathrm{u}_{\mathrm{j}^*})}{\sum_{\mathrm{j}=1}^{V} \exp(\mathrm{u}_{\mathrm{j}})} \tag{18}$$

To train the model, we minimize the cross-entropy loss function, which measures

UPDATE WEIGHTS - HIDDEN-OUTPUT LAYERS II

the difference between the true distribution (a one-hot vector centered at j^*) and the predicted probability distribution:

$$E = -\log y_{j^*} \tag{19}$$

$$= -u_{j^*} + \log \sum_{j=1}^{V} \exp(u_j) \tag{20}$$

where u_{j^*} is the logit (pre-softmax activation) corresponding to the correct output word.

This loss function represents a special case of the cross-entropy measurement between two probabilistic distributions: the true one-hot distribution and the predicted softmax probabilities.

[4] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. nature, 323(6088):533-536, 1986.

WHY log

- $\log p(x)$ is well-scaled.
- Selection of step size is easier.
- With $p(x)$, multiplication may lead to near-zero values, causing *underflow*.
- For better optimization, $\log p(x)$ is preferred because multiplication transforms into addition.

UPDATE WEIGHTS (HO) - MINIMIZATION OF E

To minimize E,take the partial derivative of E with respect to j^{th} unit of u_j

$$\frac{\partial E}{\partial u_j} = y_j - t_j = e_j \tag{21}$$

where e_j is the prediction error. Taking partial derivative with respect to the hidden-output weights, we get,

$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial w'_{ij}} = e_j \cdot h_i \tag{22}$$

Using the above equation (22),

$$w'^{new}_{ij} = w'^{old}_{ij} - \eta e_j \cdot h_i \text{ or} \tag{23}$$

$$\mathbf{v}^{(new)}_{w_j} = \mathbf{v}'^{(old)}_{w_j} - \eta e_j \cdot \mathbf{h} \quad \text{for } j = 1, 2, 3, \ldots, V \tag{24}$$

UPDATE INPUT TO HIDDEN WEIGHTS

Taking the derivative with respect to h_i, we get

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_i} = \sum_{j=1}^{V} e_j \cdot w'_{ij} = \mathbf{EH_i} \tag{25}$$

Taking the derivative with respect to w_{ki}, we get

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial h_i} \cdot \frac{\partial h_i}{\partial w_{ki}} = \mathbf{EH_i}.x_k \tag{26}$$

Now the weights are updated using

$$v_{w_i}{}^{(new)} = v_{wi}{}^{(old)} - \eta \mathbf{EH}^{\mathbf{T}} \tag{27}$$

SOME INSIGHTS ON OUTPUT-HIDDEN-INPUT LAYER WEIGHT UPDATES

- The prediction error E propagates the weighted sum of all words in the vocabulary to every output vector v'_j
- The change in the input vector is defined by the output vector which in turn is updated due to the prediction error
- The model parameters accumulate the changes until the system reaches a state of equilibrium
- The rows in the Input-Hidden layer (v_j) stores the features of the words in the vocabulary V

MATRIX OPERATIONS

x	1	0	0	0	0	0	0	0	0	0
W	0.32	0.72	0.31	0.55	0.38	0.18	0.96	0.02	0.55	0.05
	0.25	0.90	0.61	0.01	0.23	0.91	0.75	0.71	0.22	0.56
	0.16	0.31	0.61	0.51	0.71	0.96	0.31	0.24	0.90	0.63
	0.66	0.88	0.15	0.47	0.68	0.76	0.37	0.69	0.40	0.94
	0.29	0.60	0.59	0.93	0.41	0.35	0.19	0.70	0.87	0.72

$h_{in} = W^T x$		0.32		0.72		0.31		0.05		0.32
		0.25		0.90		0.61		0.56		0.25
	$1\times$	0.16	$+0\times$	0.31	$+0\times$	0.61	…	0.63	=	0.16
		0.66		0.88		0.15		0.94		0.66
		0.29		0.60		0.59		0.72		0.29

MATRIX OPERATIONS

W'										$\sigma(hin)$
0.67	0.32	0.46	0.61	0.02	0.85	0.04	0.69	0.58	0.65	0.58
0.56	0.96	0.36	0.81	0.62	0.49	0.99	0.15	0.41	0.35	0.56
0.34	0.27	0.32	0.84	0.29	0.51	0.24	0.56	0.42	0.59	0.54
0.67	0.15	0.98	0.26	0.23	0.91	0.92	0.54	0.86	0.06	0.66
0.21	0.77	0.39	0.05	0.61	0.69	0.52	0.83	0.17	0.68	0.57

$y^T = W(W^T x)$	0.09	0.09	0.10	0.09	0.06	0.17	0.11	0.11	0.09	0.08
t	0	0	0	1	0	0	0	0	0	0
e	0.09	0.09	0.10	-0.91	0.06	0.17	0.11	0.11	0.09	0.08

cross entropy loss $= 0.33$

- Every element of the $y_{j^*} > 0$ and only one element of $t_j = 1$ for $j = j^*$. $e = y_j - t_j$
- If $e > 0$, then it is over estimnated it
- $e < 0$ only when when $t_j = 1$. in such case, it is underestimated
- if $e \approx 0$, the the learning is complete and word vectors are learned. Now, the input vector is closer to the target vector.
  - In the case oc CBOW model, the context vectors are now similar to the target vector
  - In the case of skipgram model, the input word found its simialr words

MATRIX OPERATIONS

$$matmul(e, h)$$

-0.53	0.05	0.06	0.05	0.03	0.10	0.06	0.06	0.05	0.05
-0.51	0.05	0.06	0.05	0.03	0.10	0.06	0.06	0.05	0.04
-0.49	0.05	0.05	0.05	0.03	0.09	0.06	0.06	0.05	0.04
-0.60	0.06	0.07	0.06	0.04	0.11	0.07	0.07	0.06	0.05
-0.52	0.05	0.06	0.05	0.03	0.10	0.06	0.06	0.05	0.05

$$W'^{new} = W'^{old} - \eta eh$$

0.667486	0.320777	0.461016	0.608380	0.023106	0.852884	0.041133	0.692782	0.583753	0.649416
0.667486	0.320777	0.461016	0.608380	0.023106	0.852884	0.041133	0.692782	0.583753	0.649416
0.555091	0.956016	0.364246	0.811703	0.615640	0.488893	0.992000	0.150837	0.408978	0.348874
0.341566	0.266564	0.322560	0.843457	0.292134	0.510061	0.236577	0.559690	0.424626	0.594844
0.671457	0.145615	0.982863	0.259544	0.227199	0.907842	0.920170	0.542782	0.858919	0.055786
0.207933	0.767069	0.393739	0.050281	0.614529	0.687904	0.519688	0.834847	0.169773	0.679568

Word2Vec Demo

GloVe[5]

[5] Glove: Pennington, Jeffrey and Socher, Richard and Manning, Christopher Booktitle, "Global vectors for word representation", Proceedings of the conference on EMNLP, 2014

GLOVE: GLOBAL VECTORS FOR WORD REPRESENTATION

GloVe learns word embeddings by capturing global co-occurrence statistics from a corpus. Unlike Word2Vec, it optimizes word vectors using the following weighted least squares objective:

$$J = \sum_{i,j=1}^{V} f(X_{ij}) \left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij} \right)^2 \tag{28}$$

where X_{ij} is the co-occurrence count of words i and j, and $f(X_{ij})$ is a weighting function. This method effectively encodes meaning and outperforms other models in word analogy and similarity tasks.

1. Word Meaning is Encoded in Co-occurrence Ratios

- The co-occurrence count X_{ij} represents how often word j appears in the context of word i.
- The probability of co-occurrence is given by:

$$P_{ij} = \frac{X_{ij}}{\sum_k X_{ik}} \tag{29}$$

- Word relationships are better captured by ratios of these probabilities:

$$\frac{P_{ik}}{P_{jk}} = \frac{X_{ik}}{X_{jk}} \tag{30}$$

KEY FINDINGS OF THE GLOVE MODEL II

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

2. GloVe Optimizes a Weighted Least Squares Objective

- The objective function ensures that word vectors encode these probability ratios:

$$J = \sum_{i,j=1}^{V} f(X_{ij}) \left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij} \right)^2 \tag{31}$$

- The function $f(X_{ij})$ is a weighting function to handle rare and frequent words.

3. GloVe Outperforms Previous Models

- Achieves superior results on word analogy, similarity, and named entity recognition tasks.
- Captures both syntactic and semantic relationships effectively.

FastText[6]

[6] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov, "Enriching word vectors with subword information", CoRR abs/1607.04606. http://arxiv.org/abs/1607.04606

KEY FINDINGS OF THE FASTTEXT MODEL I

1. FastText Uses Subword Information

- Unlike Word2Vec and GloVe, FastText represents words as a sum of character n-grams.
- Each word is decomposed into overlapping n-grams

  Example - "apple" → {"<ap", "app", "ppl", "ple", "le>"}).
- This allows better representation of rare and morphologically rich words.

2. FastText Extends the Skip-gram Model

- The word vector is computed as the sum of its n-gram embeddings:

$$\mathbf{w} = \sum_{g \in \mathrm{G}(w)} \mathbf{z}_g \tag{32}$$

  where $\mathrm{G}(w)$ is the set of n-grams for word w and $\mathbf{z}_g$ is the embedding for each n-gram.
- Uses a negative sampling objective similar to Word2Vec for training.

3. Advantages Over Other Models

- *Handles Out-of-Vocabulary (OOV) words* by composing embeddings from subwords.
- Better for morphologically rich languages where word variations are common.
- *Faster and more efficient* for large-scale text classification.

FASTTEXT: SUBWORD TOKENIZATION OF "TECHNICAL"

FastText represents words as a sum of *character n-grams*, allowing better handling of rare and out-of-vocabulary words.

Example: Word = "technical"

Trigrams (3-grams):

- `<te`, `tec`, `ech`, `chn`, `hni`, `nic`, `ica`, `cal`, `al>`

Quadgrams (4-grams)

- `<tec`, `tech`, `echn`, `chni`, `hnic`, `nici`, `ical`, `cal>`

Key Points

- The special symbols < and > indicate *word boundaries*.
- Each n-gram gets a separate vector, and the final word embedding is the sum of its subword vectors.
- This approach allows FastText to handle *morphologically rich languages* and *unseen words*.

FASTTEXT: COMBINING SUBWORD VECTORS I

Each word is broken into *character-level n-grams* (e.g., 3-grams, 4-grams). During training, FastText stores embeddings for each subword n-gram. During inference, when a new word is encountered, it is broken into subwords using the same process. Then the word embedding is computed as

$$\mathbf{w} = \sum_{g \in \mathrm{G}(w)} \mathbf{z}_g \tag{33}$$

where

- $\mathbf{w}$ is the final word embedding.
- $\mathrm{G}(w)$ is the set of all n-grams (subwords) of w, including the whole word.
- $\mathbf{z}_g$ is the embedding of each subword g.

Example: Word = "technical"
Trigrams (3-grams)

$$G(w) = \{\texttt{<te}, \texttt{tec}, \texttt{ech}, \texttt{chn}, ..., \texttt{cal}, \texttt{al>}\}$$

Final Word Vector

$$\mathbf{w}_{\text{technical}} = \mathbf{z}_{\texttt{<te}} + \mathbf{z}_{\texttt{tec}} + \mathbf{z}_{\texttt{ech}} + ... + \mathbf{z}_{\texttt{cal}} + \mathbf{z}_{\texttt{al>}} + \mathbf{z}_{\texttt{technical}} \tag{34}$$

Key Points

- Even unseen words can be represented by summing known subword vectors.
- FastText is effective for morphologically rich languages and handling out-of-vocabulary words.

Thank you