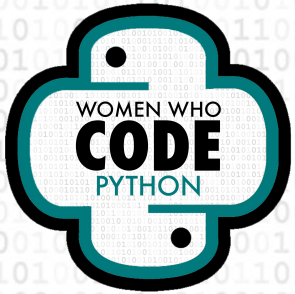
Welcome everyone!

- You can find these slides on GitHub here: https://github.com/WomenWhoCode/WWCodePython
- Please make sure your chat is set to “All panelists and attendees”.
- Some housekeeping rules:
  - Everyone will be muted throughout the webinar, but there will be opportunities for participation!
  - Please share your thoughts on the chat and/or ask questions in the Q&A.
  - The entire team is here today. Please reach out to us with any technical questions!



Discover NLP with Python

Welcome to Session #3!

THANK YOU TO OUR LEADERS!

WOMEN WHO CODE®

OUR MISSION

Inspiring women to excel in technology careers.

WOMEN WHO CODE

OUR VISION

A world where women are representative as technical executives, founders, VCs, board members and software engineers.

WOMEN WHO CODE

OUR TARGET

Engineers with two or more years of experience looking for support and resources to strengthen their influence and levelup in their careers.

WOMEN WHO CODE

CODE OF CONDUCT

WWCode is an inclusive community, dedicated to providing an empowering experience for everyone who participates in or supports our community, regardless of gender, gender identity and expression, sexual orientation, ability, physical appearance, body size, race, ethnicity, age, religion, socioeconomic status, caste, creed, political affiliation, or preferred programming language(s).

Our events are intended to inspire women to excel in technology careers, and anyone who is there for this purpose is welcome. We do not tolerate harassment of members in any form. Our **Code of Conduct** applies to all WWCode events and online communities.

Read the full version and access our incident report form at **womenwhocode.com/codeofconduct**

WOMEN WHO CODE

230,000
Members

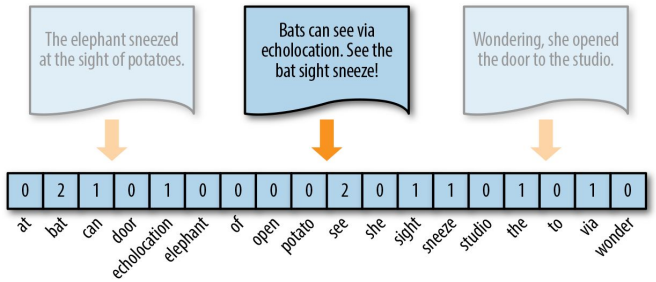
70 networks in 20 countries
Members in 97+ countries
10K+ events
$1025 daily Conference tickets
$2M Scholarships
Access to jobs + resources
Infinite connections

WOMEN WHO CODE

OUR MOVEMENT

As the world changes, we can be a connecting force that creates a sense of belonging while the world is being asked to isolate.

WOMEN WHO CODE



Word Vectorization

Word Embeddings & Vectorization and various techniques, including N-grams, Bag-of-Words, TFi-IDF, and Word2Vec

WOMEN WHO CODE PYTHON

AGENDA

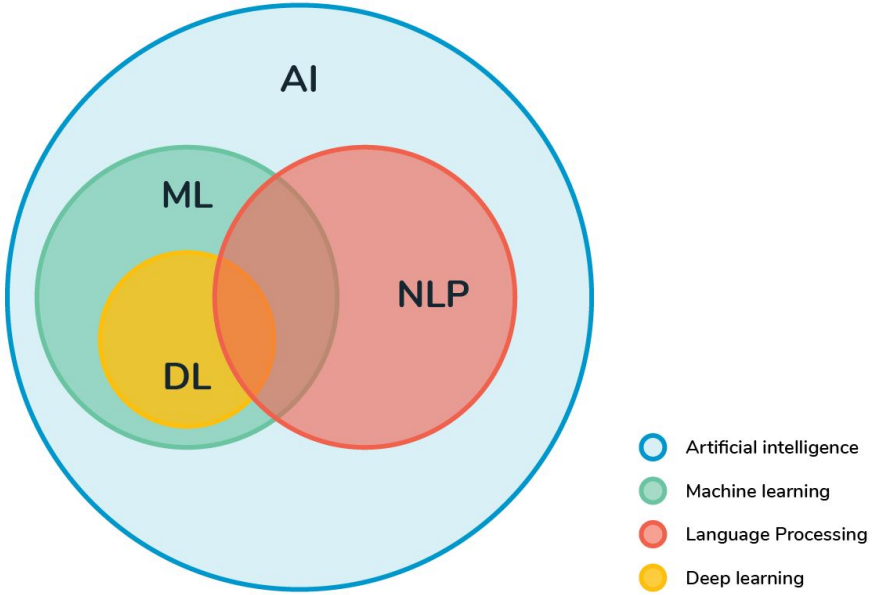
1. Recap: Sessions #1-2
2. Feature Engineering
3. N-grams
4. Bag-of-Words
5. TF-IDF
6. Word2Vec
7. Google Colab Demo
8. Recap & Wrap-up

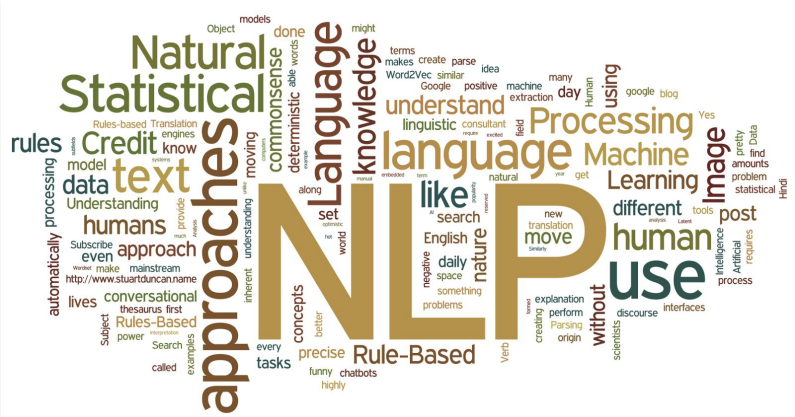
1.

Recap: Sessions #1-2

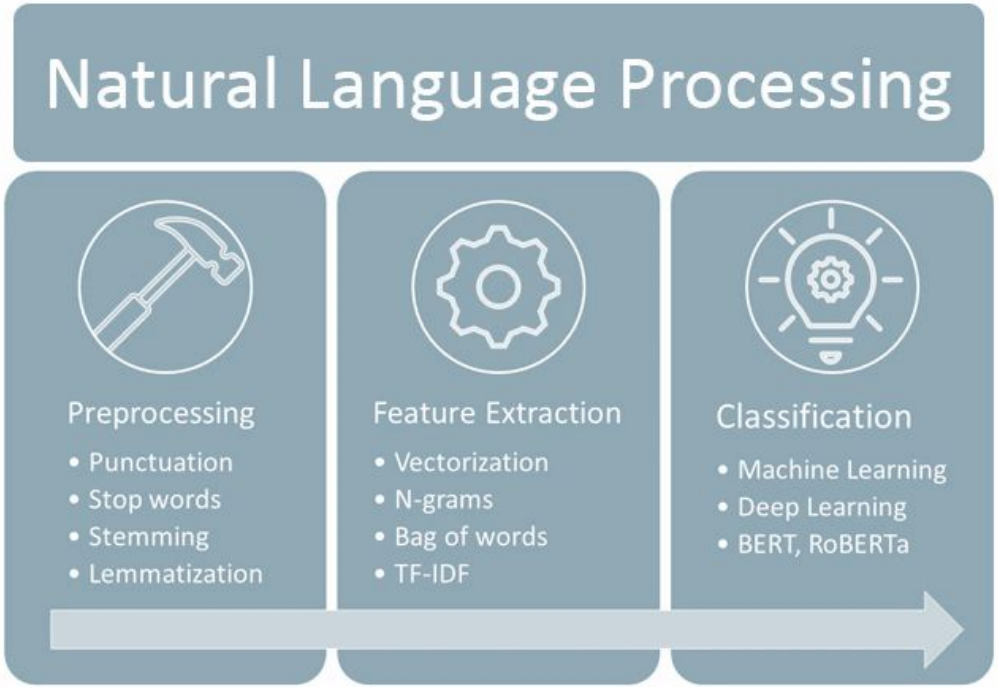
WOMEN WHO CODE PYTHON

Natural Language Processing



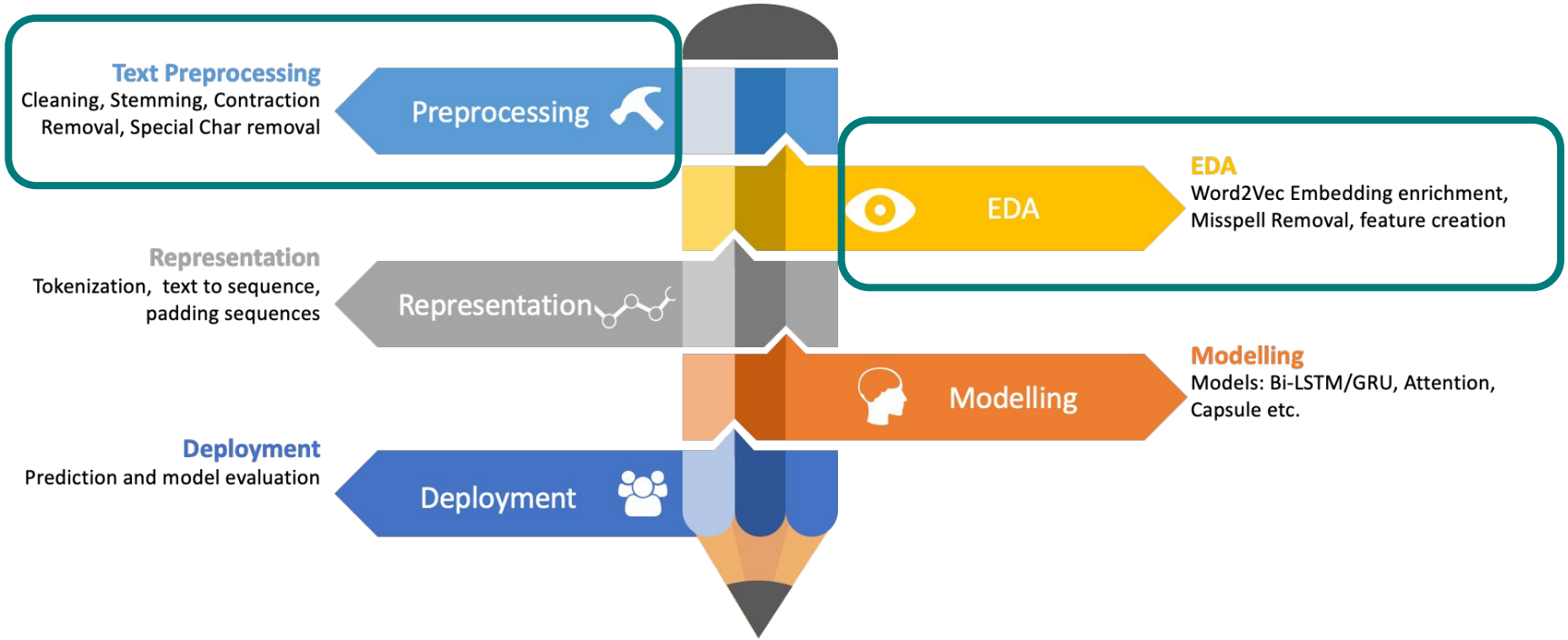


So far...



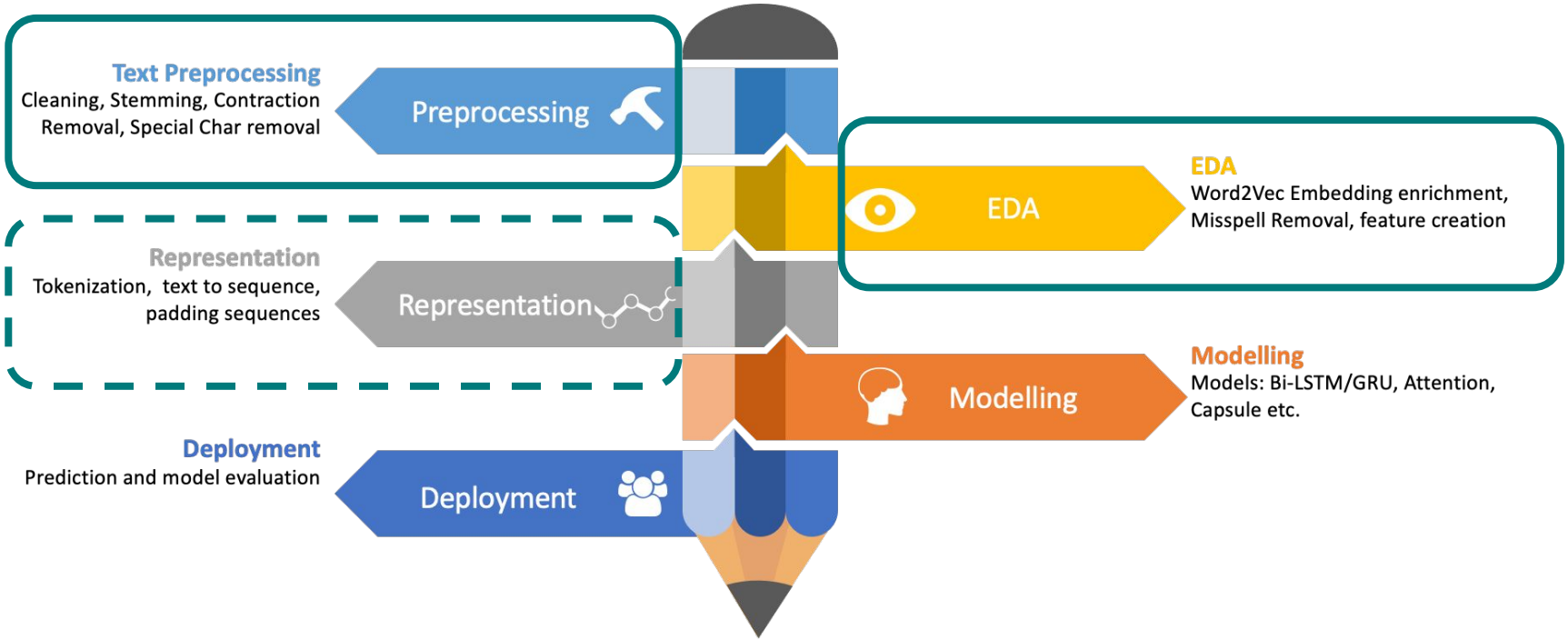
Source: https://www.kdnuggets.com/2019/10/introduction-natural-language-processing.html

The NLP Pipeline



Source: https://www.kdnuggets.com/2019/10/introduction-natural-language-processing.html

The NLP Pipeline



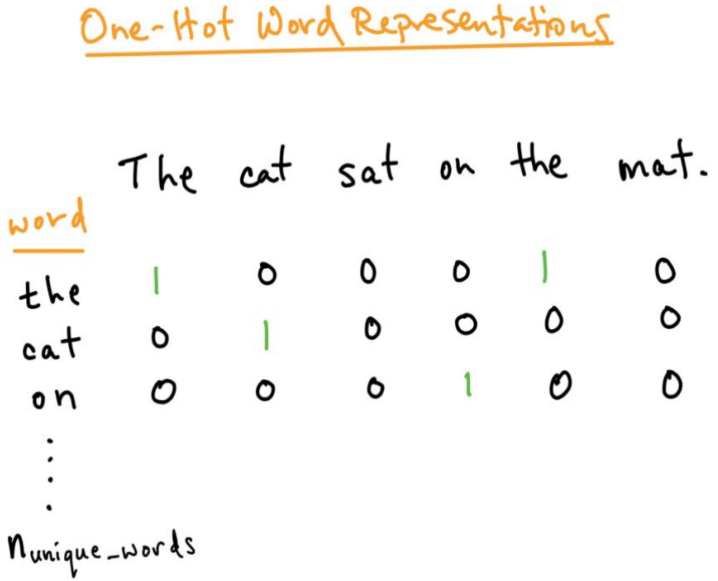
Source: https://www.kdnuggets.com/2019/10/introduction-natural-language-processing.html

2.

Feature Engineering

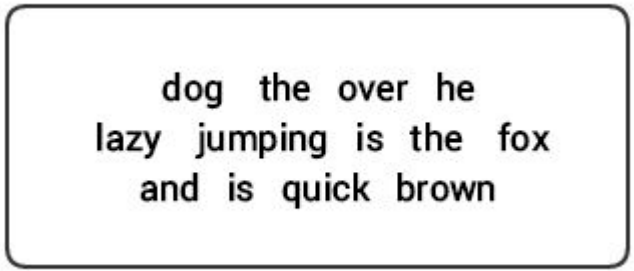
Feature Engineering Overview

- ML algorithms cannot work on the raw text directly
- Algorithms can only process **numeric representation** of an actual text
- These techniques used to convert text into a matrix (or vector)
- Popular methods of feature extraction are: Bag-of-Words, TF-IDF



Source: https://www.kdnuggets.com/2019/10/introduction-natural-language-processing.html

Understanding Syntax & Structure



- Syntax and structure are co-dependent
- A set of specific rules, conventions, and principles govern the way words are combined
- English language constituents include: words, phrases, clauses, and sentences
- Unordered words do not convey much information
- Primary techniques to understand natural language
    - Syntactic analysis (syntax) and
    - Semantic analysis (semantic)

Syntactic Vs. Semantic Analysis

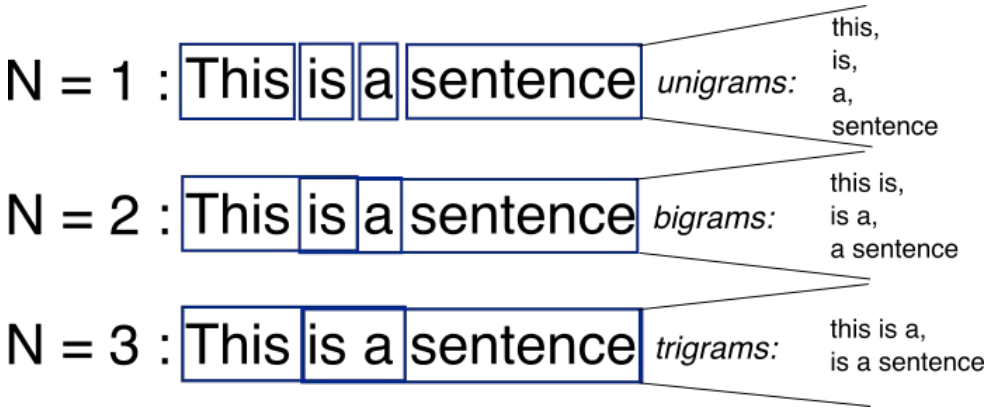
- Syntax is the grammatical structure of the text, semantics is the meaning conveyed
- Sentence that is syntactically correct, may not always be semantically correct
- Syntactic analysis basically assigns a semantic structure to text
- Semantic analysis is the process of understanding the meaning and interpretation of words, signs and sentence structure
- Ex, “cats flow supremely” is grammatically valid (subject—verb—adverb) but it does not make any sense

3.

N-Grams

WOMEN WHO CODE PYTHON

What are n-grams?



- Type of Language Model that finds the probability distribution over word sequences
- N-gram = sequence of N words
- Cut out the noise from the data
- Identify themes quickly
- NLP applications including speech recognition, machine translation and predictive text input

The Math of N-Grams: Approach #1

$P(w|h)$

Probability of word w, given some history h

Ex: P(the | today the sky is so clear that)

- w = ‘the’
- h = ‘today the sky is so clear that’

Relative frequency count

1) Take a large corpus
2) Count the number of times ‘**today the sky is so clear that’** appears
3) Count the number of times it is followed by ‘**the’**

P(the | today the sky is so clear that) =

$$\frac{C(\text{today the sky is so clear that the})}{C(\text{today the sky is so clear that})}$$

Source: https://towardsdatascience.com/introduction-to-language-models-n-gram-e323081503d9

The Math of N-Grams: Cons of Approach #1

1. If we have a large corpus, Approach 1 needs to go over entire corpus
2. Not feasible for scaling as well as time performance
3. Decomposing the probability function into smaller chunks leads to the usage of **chain rule**

Instead of computing probability using the entire corpus, chain rule using N-grams would approximate it by just a **few historical words**

The Math of N-Grams: Approach #2

P(w|h)

Probability of word w, given some history h

Ex: P(the|today the sky is so clear that)

- w_n = 'the'
- h = 'today the sky is so clear that'
- w_{n-1} = 'that'

Bigram Model

Approximates the probability of a word given all the previous words by using only the conditional probability of one preceding word

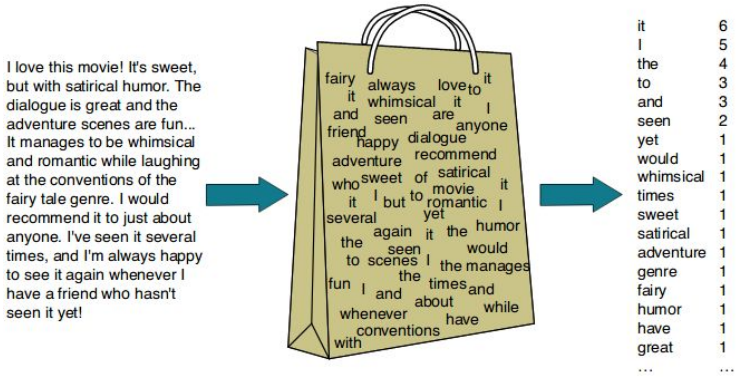
$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-1})$$

Markov assumption: Assumption that the **probability of a word** depends only **on the previous word**

Markov models are the class of probabilistic models that assume that we can **predict the probability of some future unit** without looking too far in the past

4.

Bag-of-Words



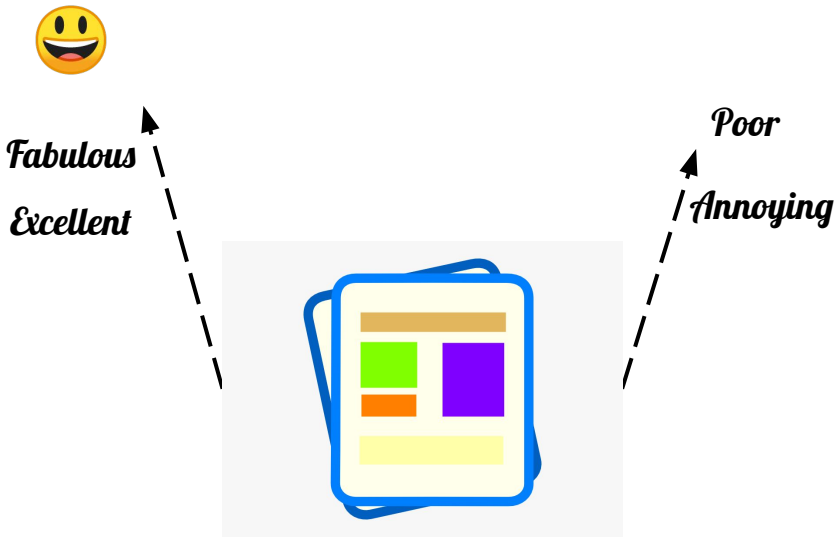
WOMEN WHO CODE PYTHON

Why a “bag”?

- Any information about the order or structure of words is discarded
- Only concerned with whether known words occur in the document, *not where*
- General intuition: similar documents have similar content

Bag-of-Words Model

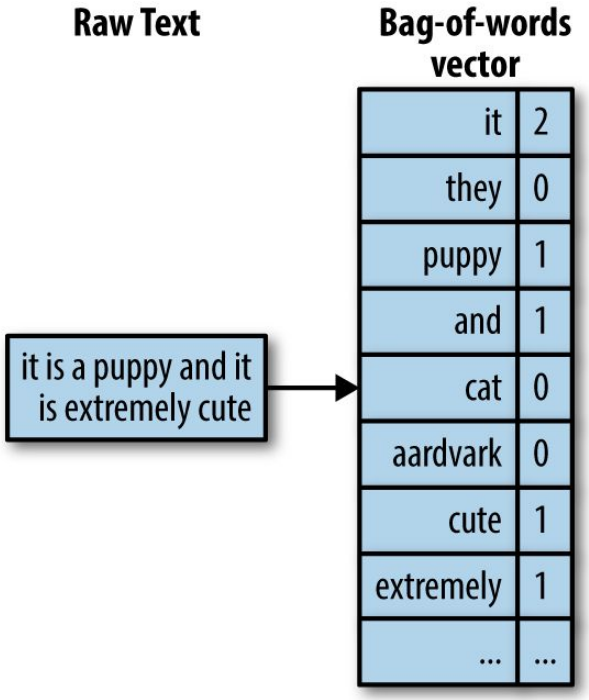
- Transform tokens into a set of features
- Used in document classification, each word used as a feature for training the classifier
- Ex: review-based sentiment analysis



Source: https://towardsdatascience.com/understanding-feature-engineering-part-3-traditional-methods-for-text-data-f6f7d70acd41

Bag-of-Words Model

- Each text document = numeric vector
- Each dimension = specific word (from the corpus)
- Value = frequency in document, occurrence (1/0), or weighted value



Source: Source: http://uc-r.github.io/creating-text-features

Bag-of-Words Model: Simple Example

“A Tale of Two Cities”
by Charles Dickens

> *It was the best of times,*
> *it was the worst of times,*
> *it was the age of wisdom,*
> *it was the age of foolishness,*

Source: https://machinelearningmastery.com/gentle-introduction-bag-words-model/

Step 1: Collect Data

- Data: first few lines of text from book
- Each line treated as a separate “document”
- Four lines = entire corpus of documents

Source: https://machinelearningmastery.com/gentle-introduction-bag-words-model/

Step 2: Design the Vocabulary

- Construct list of all words in the model's vocabulary
- *Only* the *unique* words (note: we ignore case and punctuation)
- Our list: vocab of 10 words, from corpus of 24 words

- "it"
- "was"
- "the"
- "best"
- "of"
- "times"
- "worst"
- "age"
- "wisdom"
- "foolishness"

Source: https://machinelearningmastery.com/gentle-introduction-bag-words-model/

Step 3: Create Document Vectors

- Score the words in each document
- Document of free text → vector
- Vector = input *or* output of model
- Vocab of 10 words → fixed-length vector
- Use 0/1 binary scoring

- “it” = 1
- “was” = 1
- “the” = 1
- “best” = 1
- “of” = 1
- “times” = 1
- “worst” = 0
- “age” = 0
- “wisdom” = 0
- “foolishness” = 0

Source: https://machinelearningmastery.com/gentle-introduction-bag-words-model/

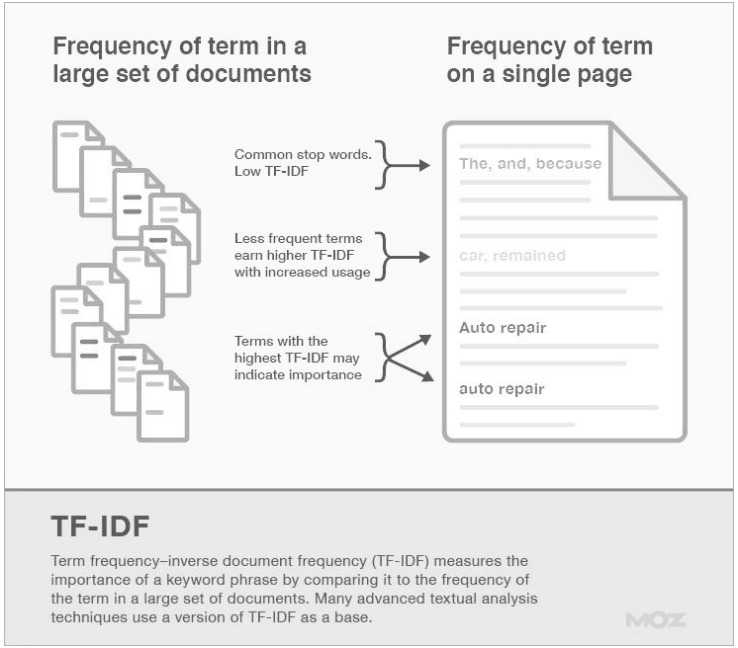
Managing Vocabulary

- Vocab size ⬆ Vector representation of documents ⬆
- Issue of sparse vector/representation
  - Memory, computational resources
- Decrease vocab size → text cleansing!
  - Ignore case
  - Ignore punctuation
  - Ignore stop words
  - Fix misspelled words
  - Stemming

Source: https://machinelearningmastery.com/gentle-introduction-bag-words-model/

5.

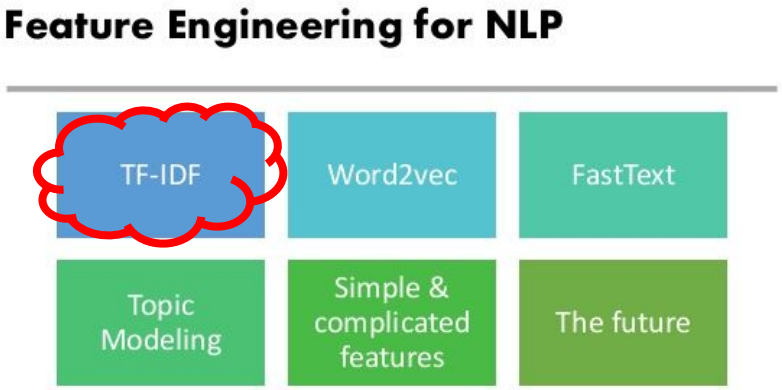
TF-IDF



WOMEN WHO CODE PYTHON

TF-IDF

- Term frequency-inverse document frequency
- Combination of two metrics: **term frequency** and **inverse document frequency**
- Commonly used in information retrieval



Source: https://www.slideshare.net/BillLiu31/feature-engineering-for-nlp

Goal of TF-IDF

- Text documents → vector models, based on word occurrence (without considering the exact ordering)
- Given: dataset of N text documents, TF and IDF defined as...
  - TF: count of a term “t” in a document “d”
  - IDF: logarithm of ratio of total documents (d) in the entire corpus and number of documents (d) containing the term “t”
- Together: relative importance of a term in a corpus

TF-IDF: Formula (1)

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

Source: http://filotechnologia.blogspot.com/2014/01/a-simple-java-class-for-tfidf-scoring.html

TF-IDF: Formula (2)

$$TF_{ij} = \frac{f_{ij}}{n_j} \quad (1)$$

Equation 1

Where f_{ij} is the frequency of term i in document j. n_j is the total number of words in document j.

$$IDF_i = 1 + log(\frac{N}{c_i}) \quad (2)$$

Equation 2

Where N is the total number of documents in the corpus. c_i is the number of documents that contain word i.

$$w_{ij} = TF_{ij} \times IDF_i \quad (3)$$

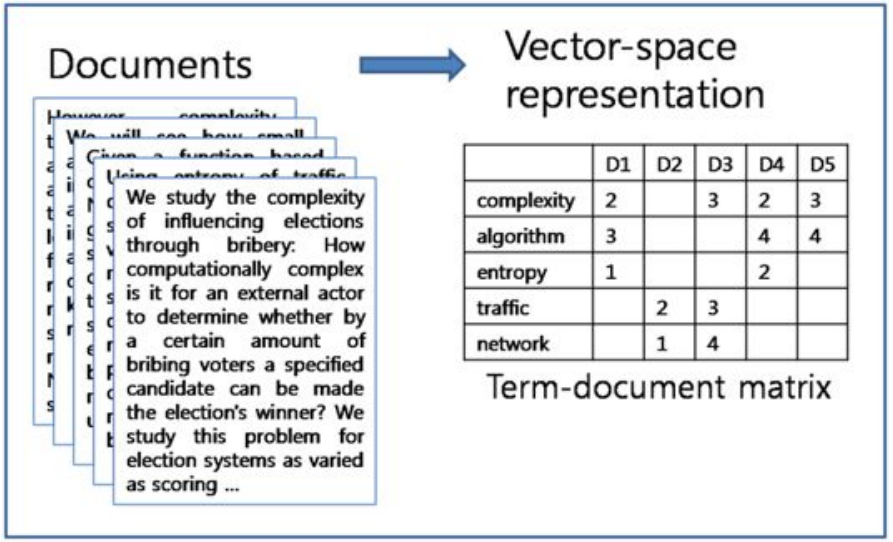
Equation 3

Where w_{ij} is the TF-IDF score of term i in document j.

Source:

TF-IDF: Properties

- Compute for every word in every document
- Result: matrix with shape = # words * # documents
- A single value for 1 word → matrix of values when considering all documents



	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

Source: https://www.quora.com/What-is-a-tf-idf-vector

TF-IDF: Simple Example

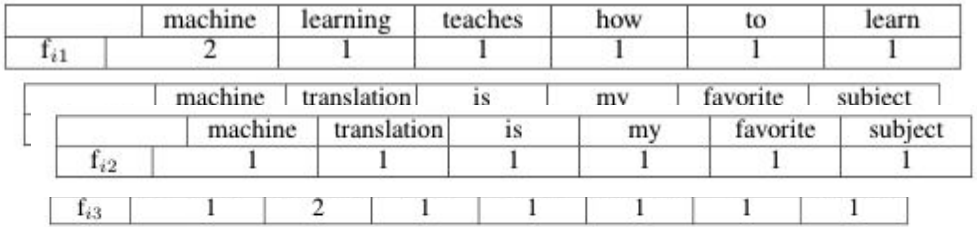
3 Documents

- Document 1: “Machine learning teaches machine how to learn”
- Document 2: “Machine translation is my favorite subject”
- Document 3: “Term frequency and inverse document frequency is important”

Source:

TF-IDF: Step 1

- Compute f_{ij}: frequency of term i in document j, Equation 1
- Pure counting: each term i in documents 1-3
  - f{i1}
  - f{i2}
  - f{i3}



	machine	learning	teaches	how	to	learn
f_{i1}	2	1	1	1	1	1

	machine	translation	is	my	favorite	subject
f_{i2}	1	1	1	1	1	1

f_{i3}	1	2	1	1	1	1	1

Source:

TF-IDF: Step 2

- Compute normalized term frequency TF_{ij}, Equation 1
- Each word in single document should divide total # words in that document

	machine	learning	teaches	how	to	learn
TF_{i1}	0.29	0.14	0.14	0.14	0.14	0.14

	machine	translation	is	my	favorite	subject
TF_{i2}	0.17	0.17	0.17	0.17	0.17	0.17

	term	frequency	and	inverse	document	is	important
TF_{i3}	0.125	0.25	0.125	0.125	0.125	0.125	0.125

Source:

TF-IDF: Step 3

- Compute IDF of each term i, Equation 2

Terms	IDF
Machine	1.4
learning	2.1
teaches	2.1
how	2.1
to	2.1
learn	2.1
translation	2.1
is	1.4
my	2.1
favorite	2.1
subject	2.1
Term	2.1
frequency	2.1
and	2.1
inverse	2.1
document	2.1
important	2.1

Source:

TF-IDF: Step 4

- Compute TF-IDF for each word i in document j, Equation 3

	machine	learning	teaches	how	to	learn
$TFIDF_{i1}$	0.40	0.30	0.30	0.30	0.30	0.30

	machine	translation	is	my	favorite	subject
$TFIDF_{i2}$	0.23	0.35	0.23	0.35	0.35	0.35

	term	frequency	and	inverse	document	is	important
$TFIDF_{i3}$	0.26	0.52	0.26	0.26	0.26	0.18	0.26

Source:

TF-IDF: A Given Query

- Given a query: “machine learning” → compute TF-IDF

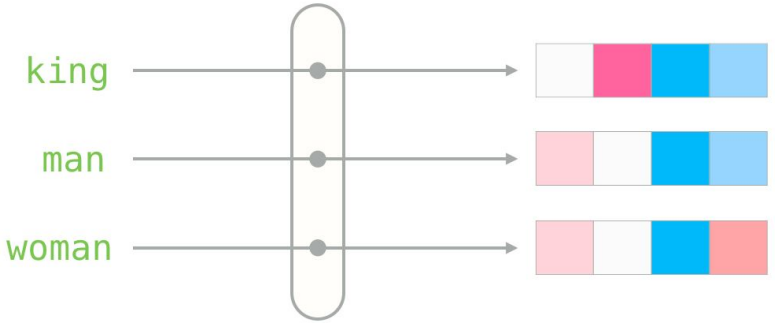
	TF	IDF	TF*IDF
machine	0.5	1.4	0.7
learning	0.5	2.1	1.05

	Document1	Document2	Document3
machine	0.4	0.23	0.0
learning	0.3	0	0.0

Source:

6.

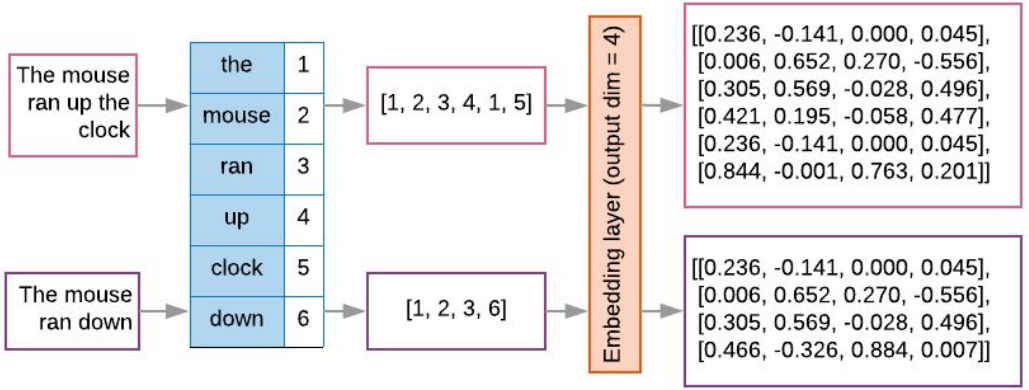
Word2Vec



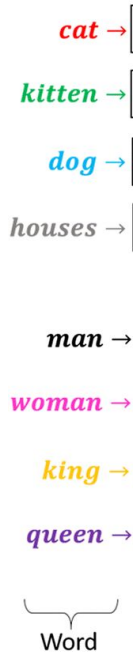
WOMEN WHO CODE PYTHON

Word Embeddings

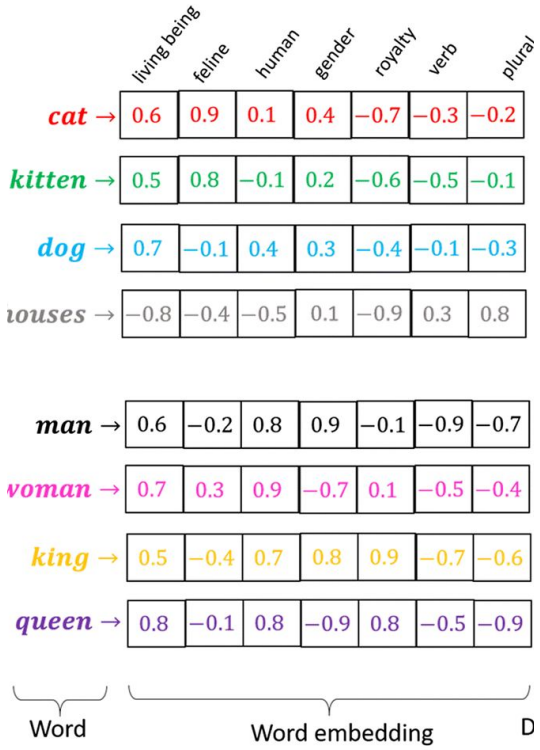
- Vector representations of a given word
  - Words with similar meaning → similar representation
- Capture properties of a word in a particular document
  - Context
  - Semantic, syntactic similarity
  - Relation to other words
- Real-valued vectors in a predefined vector space
  - Each word mapped to one vector
  - Vector values learned in a way that resembles a neural network



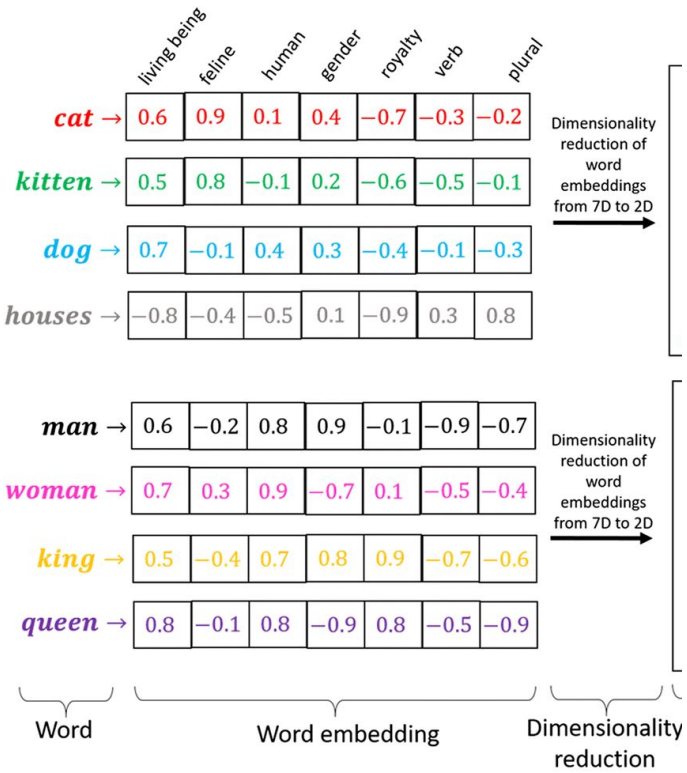
Word Embeddings: Break it Down



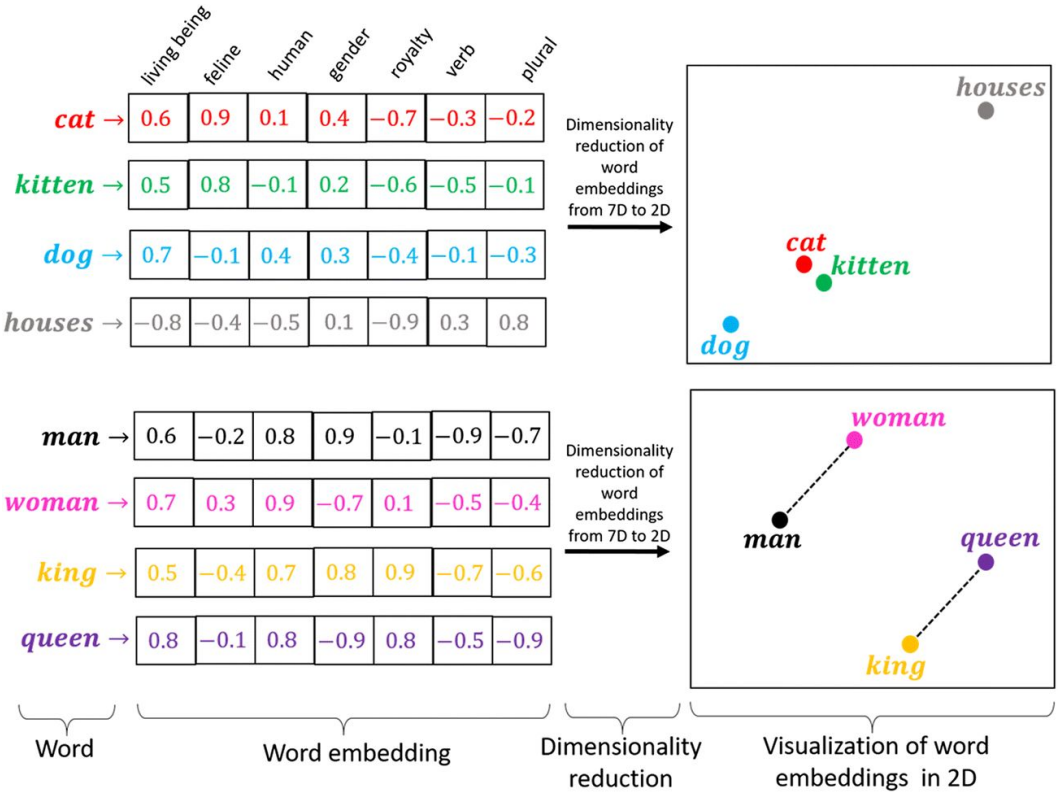
Word Embeddings: Break it Down



Word Embeddings: Break it Down

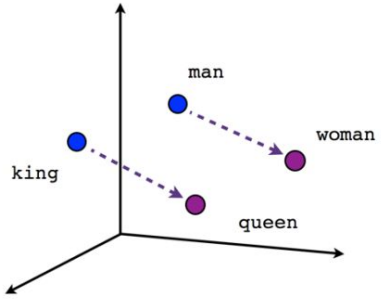


Word Embeddings: Break it Down

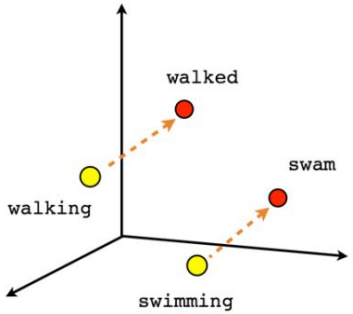


Word2Vec

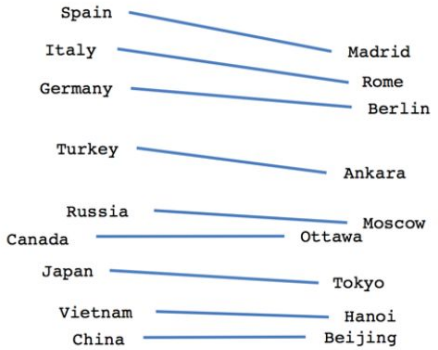
- Perhaps the most famous face of neural nets + NLP
- Uses a (skip-gram) neural network model to learn word associations (predict neighbor context) from any large corpus (text)
- Each distinct word represented as a vector
    - Carefully selected: cosine similarity (semantic similarity)



Male-Female



Verb tense

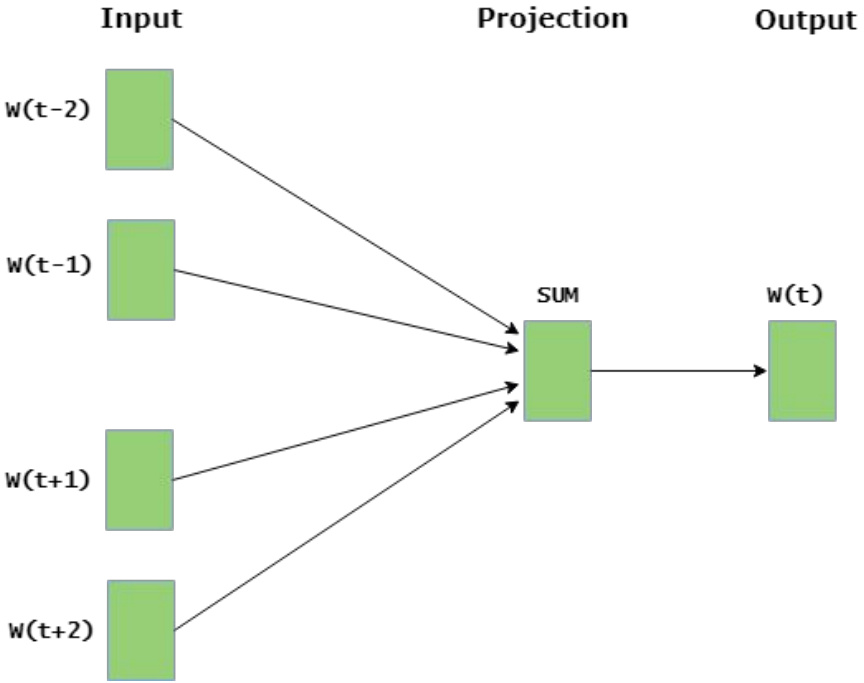


Country-Capital

Word2Vec: Architectures

Continuous Bag-of-Words (CBOW)

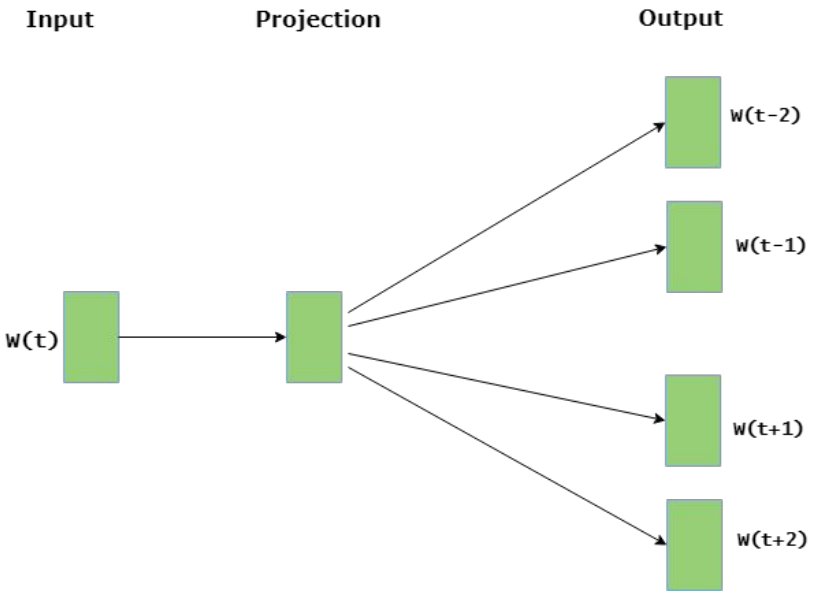
- Predict the current word given context words w/n specific window
- Input layer: context words
- Output layer: current word
- Hidden layer: # of dimensions in which we want to represent current word



Word2Vec: Architectures

Skip Gram

- Predict surrounding context words w/n specific window, given the current word
- Input layer: current word
- Output layer: context words
- Hidden layer: # dimensions in which we want to represent current word




PHEW!!

7.

Google Colab - Live Coding!

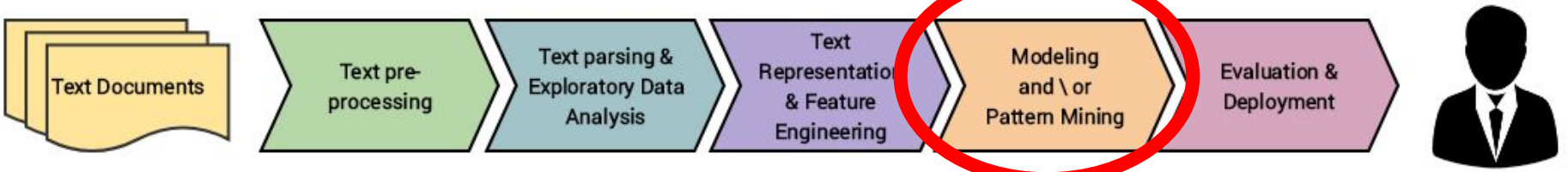
https://bit.ly/DiscoverNLP_Ses3

WOMEN WHO CODE PYTHON

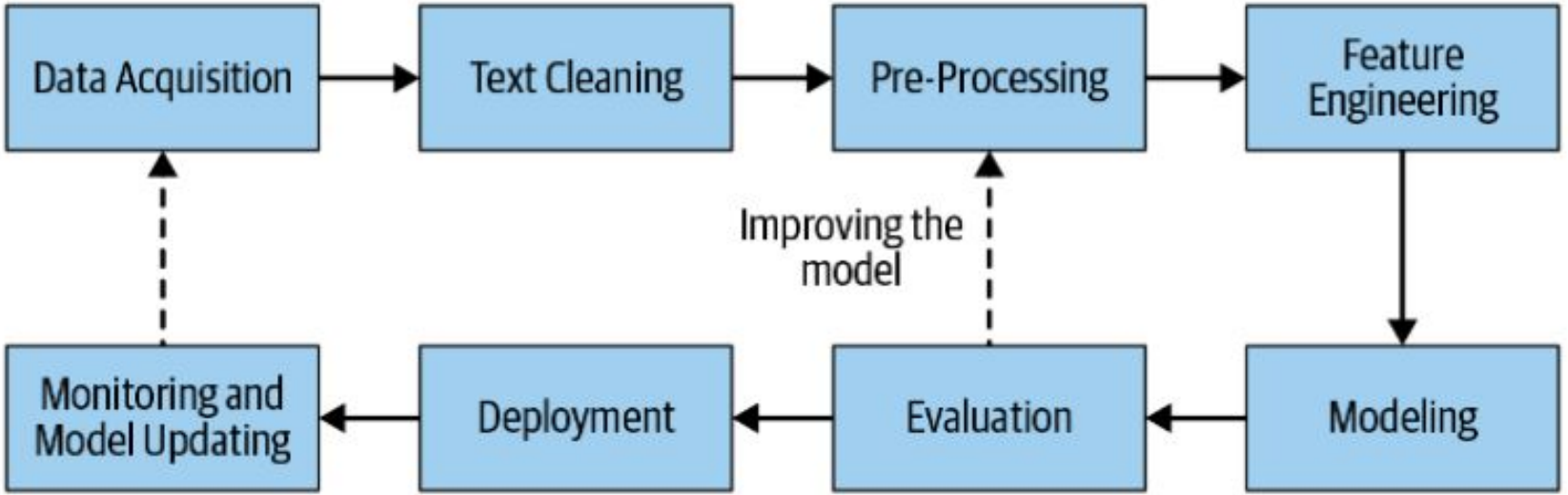
8.

Recap & Next Steps

WOMEN WHO CODE PYTHON


Text Documents
Text pre-processing
Text parsing & Exploratory Data Analysis
Text Representation & Feature Engineering
Modeling and \ or Pattern Mining
Evaluation & Deployment

The NLP Pipeline



Source: https://www.kdnuggets.com/2019/10/introduction-natural-language-processing.html

Next Session!

Neural Models for Machine Translation & Conversation

December 13th, 7:00-8:30 pm (EST)

Upcoming Events!

WED
02
DEC

✨ **Beginner Python Study Group** ✨ **Session 8: Mini Project** *Recurring*
1:00 PM – 2:30 PM (SST) | Zoom

Register

SUN
13
DEC

✨ **Discover NLP with Python Study Group: Session 4** ✨ *Featured*
1:00 PM – 3:00 PM (SST) | Zoom

Register

Stay Connected

Questions?

Join our Slack channel: #discover-nlp-with-python

Thanks everyone!