Building Robust Text Processing Pipelines

Maryam Jahanshahi Ph.D.

bit.ly/rladies2019

Text Analysis: The Final Frontier

Extracting structured insights from unstructured inputs is not easy

The amount of digital data is **doubling** every year

Less than **0.5%** of all data in a company is analyzed

More than **75%** of all data is in text form

Source: IDC iView "Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East," December 2014.

About me

Past

Journalist > Editor
on pipeline building

Cancer Biologist
on scaling sizes

Present

Research Scientist
TapRecruit

How do we make decisions about our careers?

What role do documents play in this process?

Future

Achtung!

The focus of this talk will **not** be about text analysis in R

Corpus Processing

spaCy

Language Modeling

gensim

TensorFlow

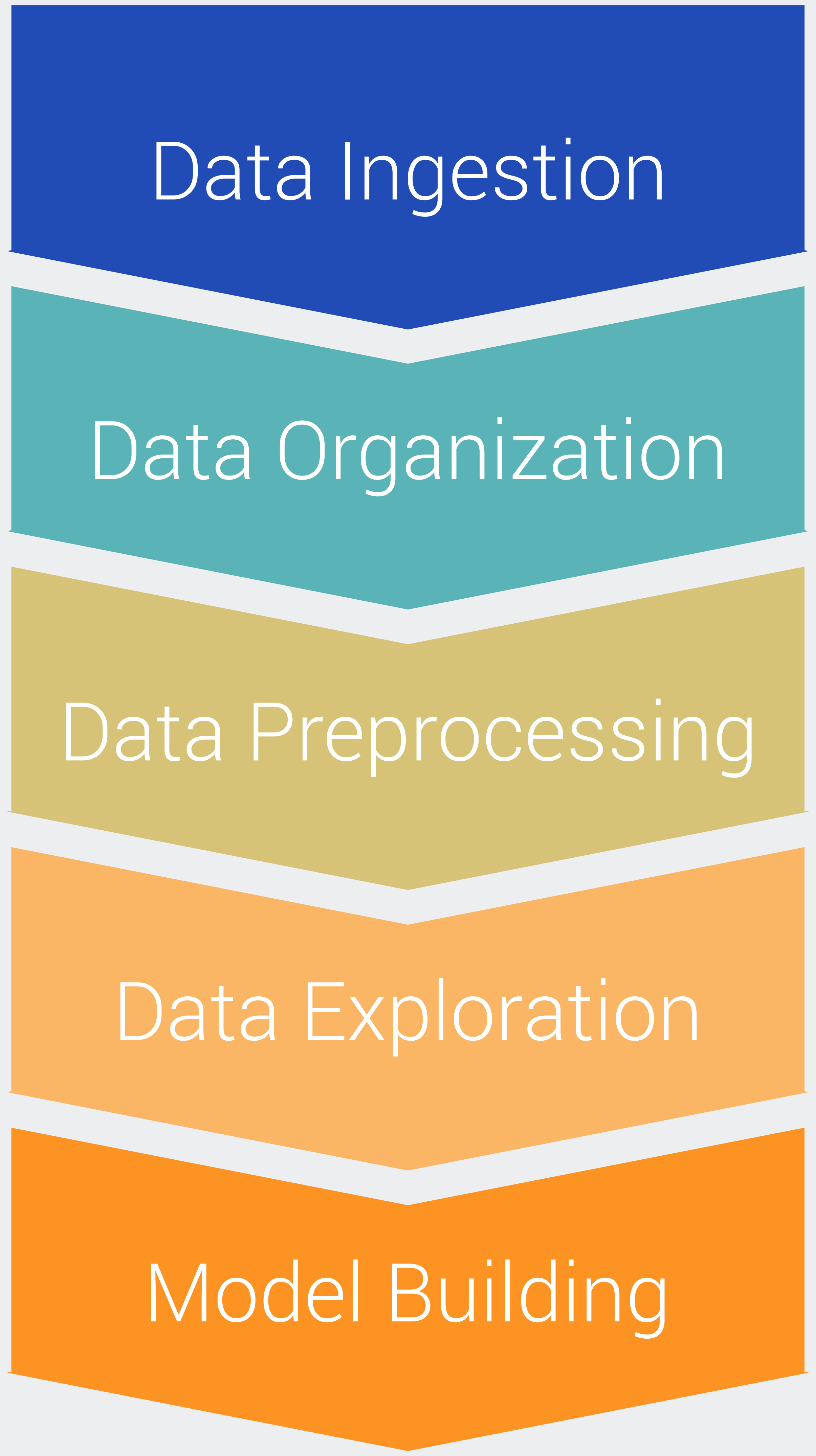
Stack Considerations

- Actively developed libraries
- Industrial-strength NLP:
  - Parallel processing of large datasets
  - Prototype to Production
  - Effective memory management

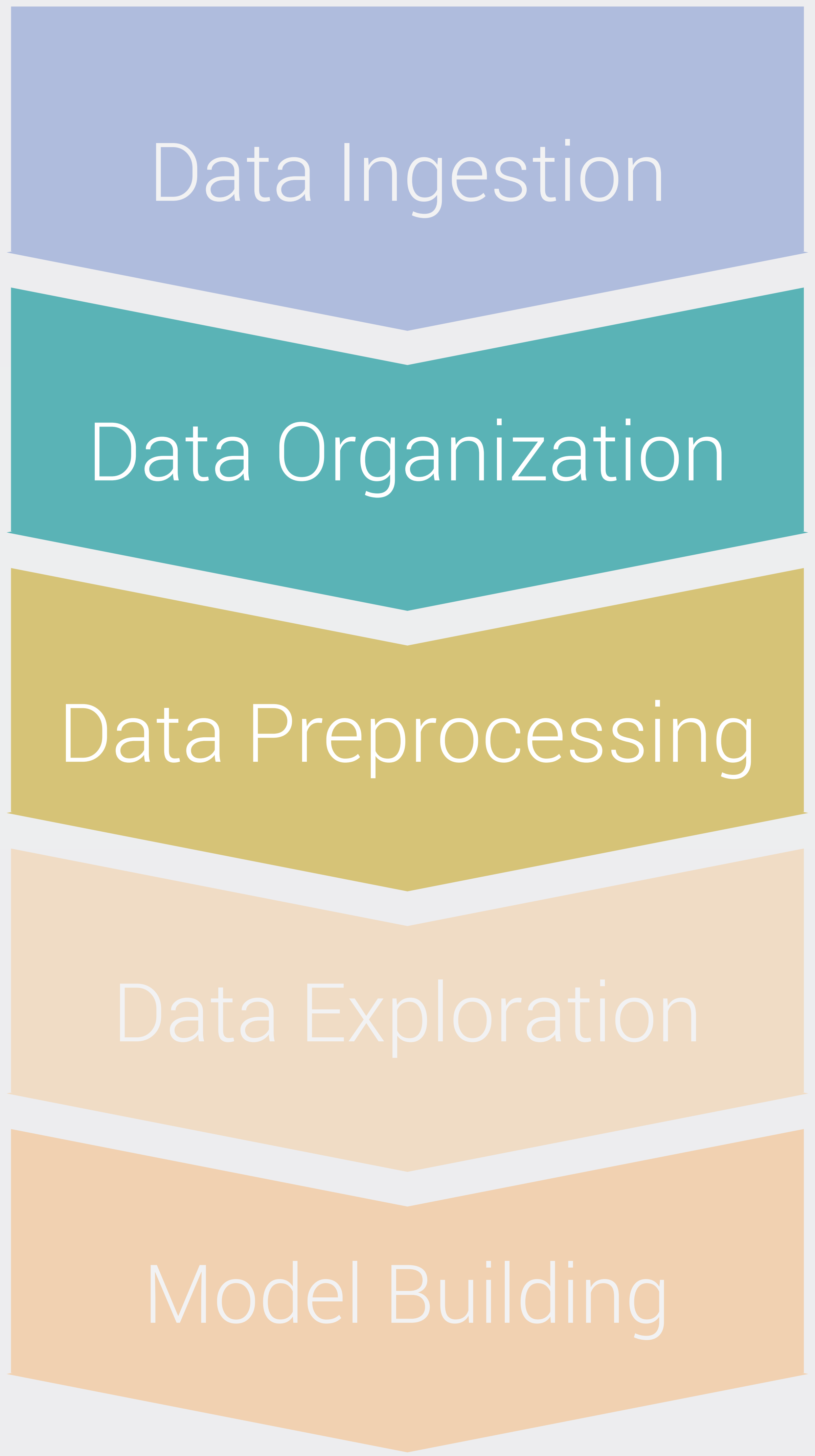
Design Considerations

- Maintainability
- Reproducibility
- Environments

A Day in the Life of an NLP Project



A Day in the Life of an NLP Project



‘Amanda didn’t start the fire’

(‘fire’, NN)

Corpus: A collection of documents

Document: Unprocessed string, typically associated with structured data

Segment: Processed string (i.e. sentence, paragraph etc.)

Token: Processed single data point

Designing Data Preprocessors

Clean up	**Goal:** Remove inconsistency between otherwise similar data points
Segmentation	**Goal:** Split text chunks into data points (i.e. the unit of analysis or evaluation)
Normalization	**Goal:** Put data points on an equal footing

Designing Data Preprocessors

Clean up

Segmentation

Normalization

General Considerations

- What is the unit or data structure of analysis? (Tokens vs sentences vs paragraphs vs docs)
- Can the cleanup aid segmentation?

Specific Considerations

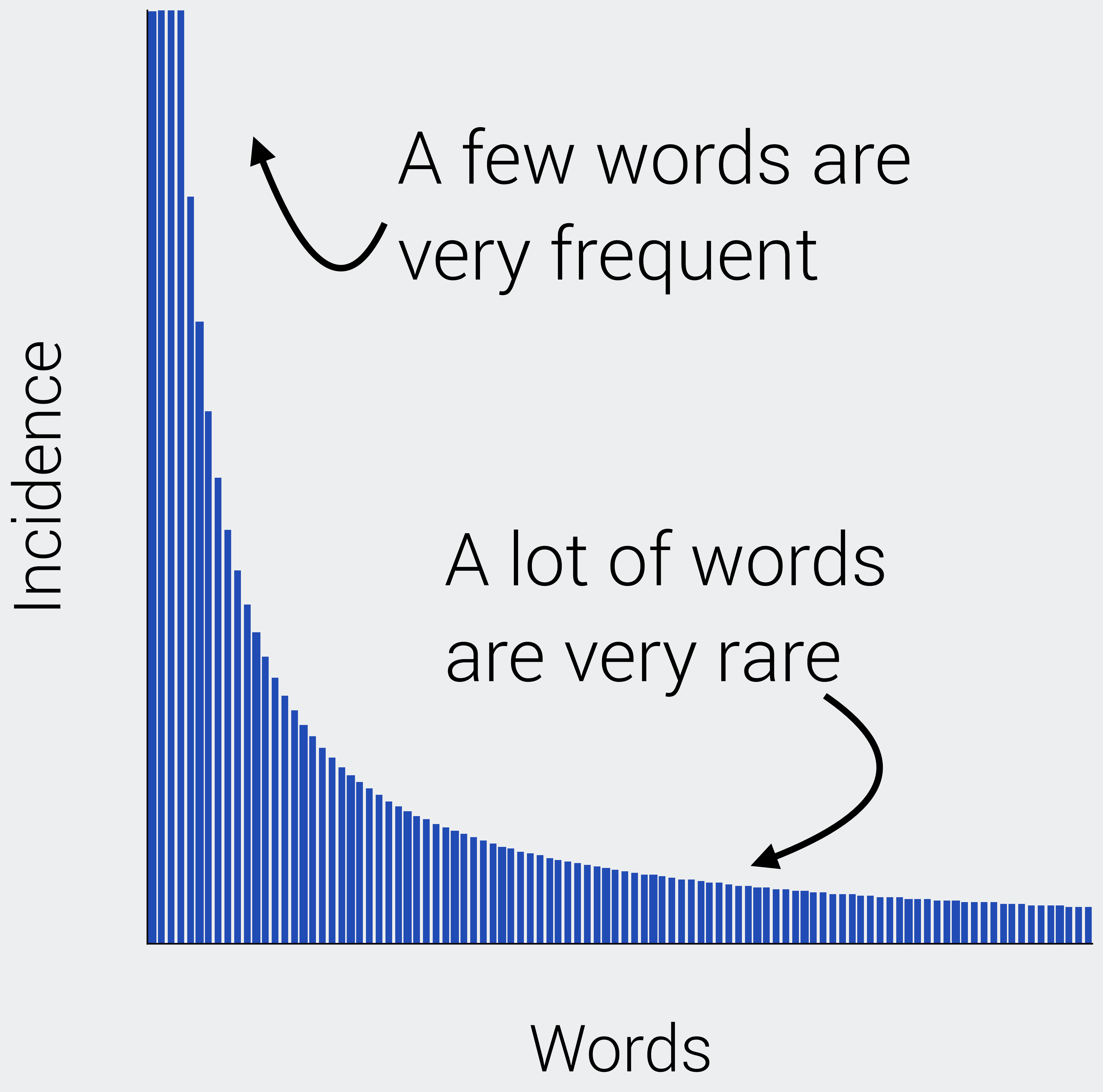
- What is the role of punctuation?
- What role do hyphenated words play?
- Will parsing emojis or emoticons be helpful?

Functions in a typical clean up script

The order of operations is important

HTML/XML	<p>Amanda didn’t start the fire!</p>	Python: **Beautiful Soup** R: **xml2?**
Unicode	<p>Amanda didn’t start the fire!</p>	Python: **regex** R: **utf8**
Contractions	Amanda did not start the fire!	Python: **spaCy** R: **textclean**
Punctuation	Amanda did not start the fire	Python: **spaCy** R: **textclean**

Word incidence is rarely distributed normally



- **Stop words**: Removing most frequent words.
  - Standard list with most NLP libraries
  - Make your own artisanal list
- **Changing cases**:
  - Standard is to convert to lower case
  - Casing may matter for you (e.g. IT vs it)
- **Process numbers**:
  - Standard is to remove all numbers
  - Convert into words via **`inflect`** library in Python and **`textclean`** package in R
- **Stem or lemmatize words**:
  - Standard is to lemmatize

Best Practices in Data Organization

```
corpus
|_README.md
|_raw
  |_01.txt
  |_02.txt
  |_03.txt
  |_metadata.json
|_processed
  |_processed.json
  |_metadata.json
|_scripts
```

Processed documents:
Save down processed documents either as JSON objects or in a document database (NoSQL)

Metadata:
Define what has been processed and when in metadata:
- Files
- Words
- Unique Tokens
- Date of latest preprocessing

Advanced Best Practices in Data Organization

Create a corpus reading module:

- Define which files should be loaded and how those files should be loaded.
    - Store these as parameters in README.
    - Regex for file names / formats [**`\w\.txt+`**]
    - Can include a filter list for restricting files

```
corpus
|_README.md
|_raw
  |_01.txt
  |_02.txt
  |_03.txt
  |_metadata.json
|_processed
  |_processed.json
  |_metadata.json
|_scripts
```

Advanced Best Practices in Data Organization

Create a corpus reading module:

- Define which files should be loaded and how those files should be loaded.
    - Store these as parameters in README.
    - Regex for file names / formats [**`\w\.txt+`**]
    - Can include a filter list for restricting files

```
import json

def project_reader(self):
    return json.load(self.open(“README”))
```

Feature Extraction

Bag of Words representation vectorizes through word counts

	amanda	baby	did	doo	fire	not	shark	start	the
Amanda didn’t start the fire	1	0	1	0	1	1	0	1	1
Baby shark, doo doo doo doo doo doo	0	1	0	6	0	0	1	0	0

Feature Extraction

Bag of Words representation vectorizes through word counts

Amanda didn’t start the fire

	amanda	baby	did	doo	fire	not	shark	start	the
Bag of Words	1	0	1	0	1	1	0	1	1
- stop words	1	0	0	0	1	0	0	1	0
+ normalization	0.3	0	0	0	0.3	0	0	0.3	0

Feature Extraction

One Hot Encoding and TFIDF normalize token frequencies

Baby shark, doo doo doo doo doo doo

	amanda	baby	did	doo	fire	not	shark	start	the
Bag of Words	0	1	0	6	0	0	1	0	0
One Hot Encoding	0	1	0	1	0	0	1	0	0
TFIDF	0	0.05	0	0.4	0	0	0.2	0	0

Feature Extraction

Transforming text data into numeric features

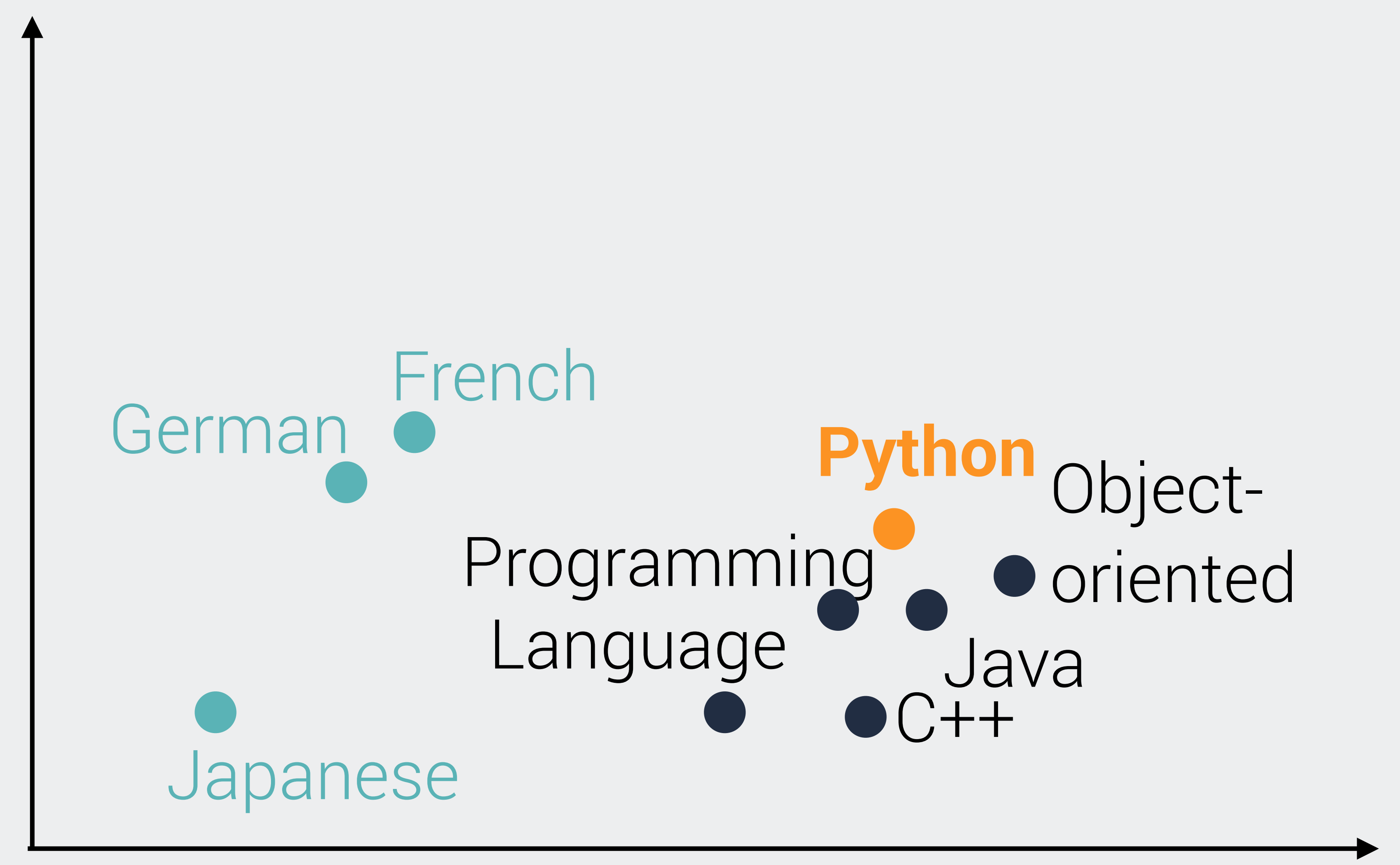
Non-distributed			Distributed
One Hot Encoding	Bag of Words	TF-IDF	Embeddings
Stop word removal?	Stop word removal	No need for stop word removal	Stop word removal?
None	Document-level normalization	Corpus and document-level normalization	Context and corpus-level normalization

Word embeddings capture semantic similarities

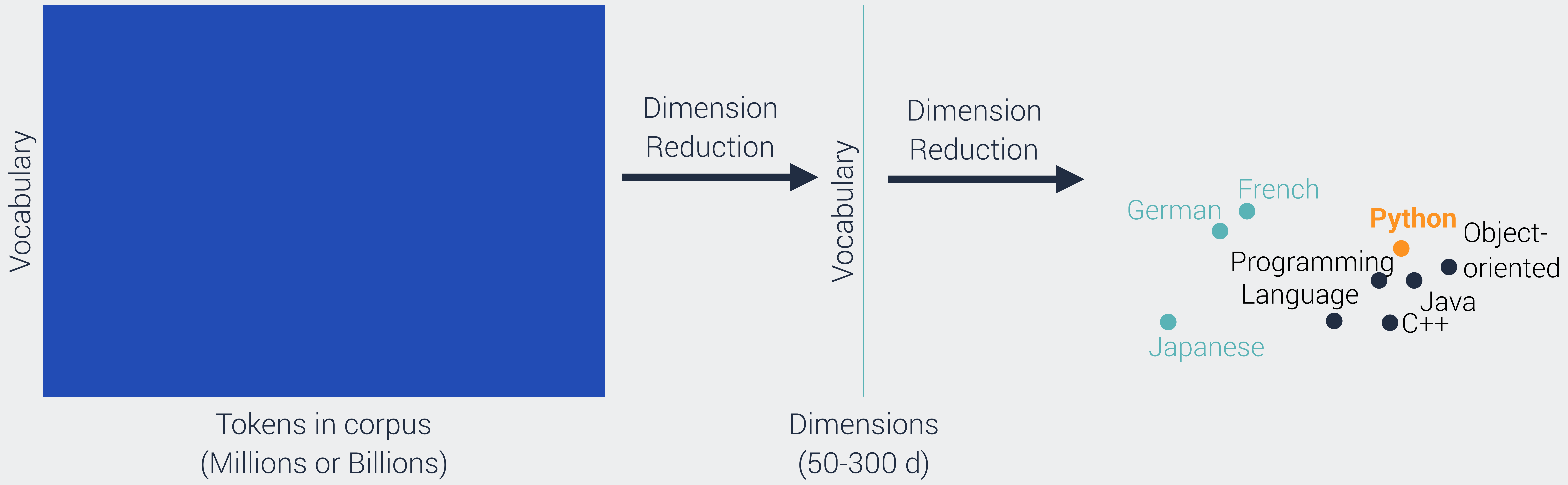
$\underbrace{\text{Statistical modeling through software (e.g. SPSS) or programming language (e.g.}}_{\text{Context}}$ $\underset{\text{Word}}{\textbf{Python}}$)

$\underbrace{\text{Experience in}}_{\text{Context}}$ $\underset{\text{Word}}{\textbf{Python}}$, $\underbrace{\text{Java or other object-oriented programming languages}}_{\text{Context}}$

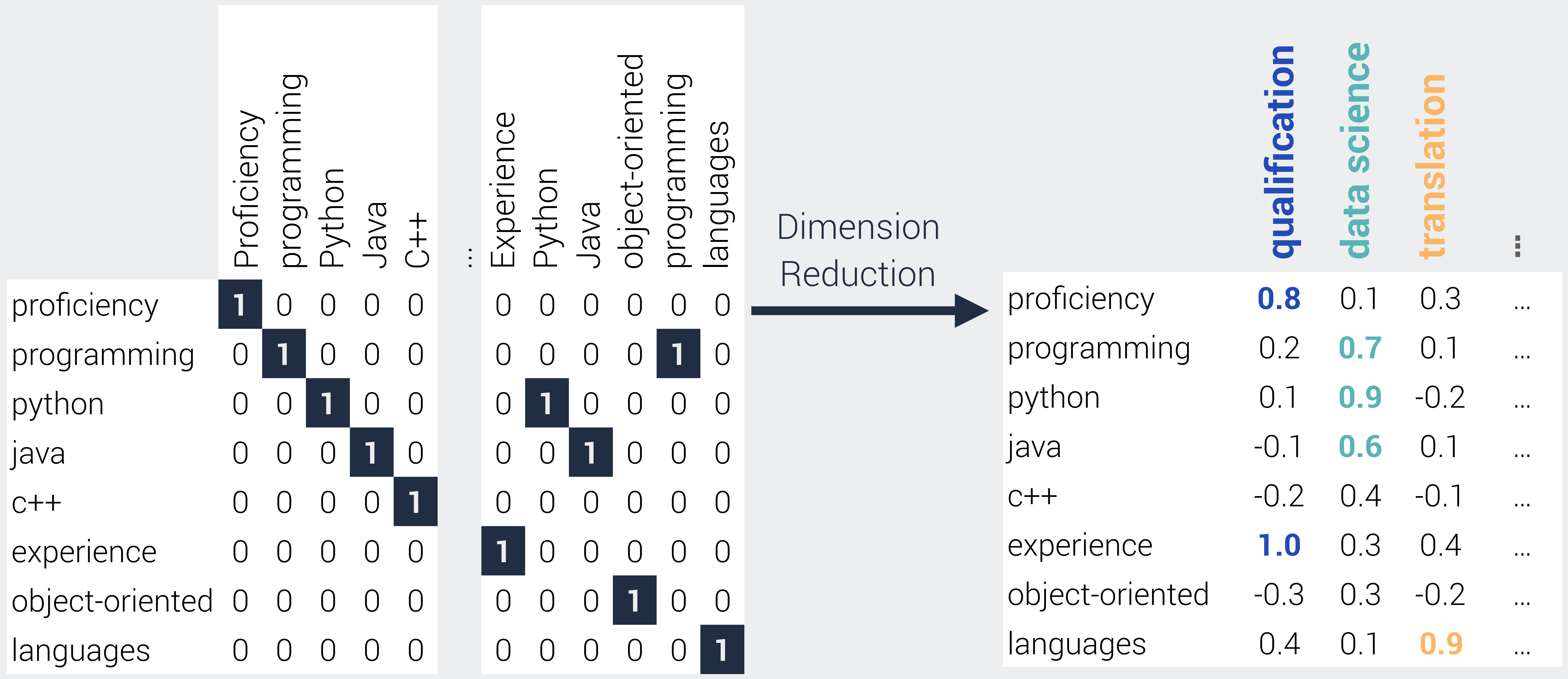
$\underbrace{\text{Proficiency programming in}}_{\text{Context}}$ $\underset{\text{Word}}{\textbf{Python}}$, $\underbrace{\text{Java or C++.}}_{\text{Context}}$



A simplified representation of word vectors



A simplified representation of word vectors



	Proficiency	programming	Python	Java	C++	⋮	Experience	Python	Java	object-oriented	programming	languages
proficiency	1	0	0	0	0		0	0	0	0	0	0
programming	0	1	0	0	0		0	0	0	0	1	0
python	0	0	1	0	0		0	1	0	0	0	0
java	0	0	0	1	0		0	0	1	0	0	0
c++	0	0	0	0	1		0	0	0	0	0	0
experience	0	0	0	0	0		1	0	0	0	0	0
object-oriented	0	0	0	0	0		0	0	0	1	0	0
languages	0	0	0	0	0		0	0	0	0	0	1

	qualification	data science	translation	⋮
proficiency	**0.8**	0.1	0.3	...
programming	0.2	**0.7**	0.1	...
python	0.1	**0.9**	-0.2	...
java	-0.1	**0.6**	0.1	...
c++	-0.2	0.4	-0.1	...
experience	**1.0**	0.3	0.4	...
object-oriented	-0.3	0.3	-0.2	...
languages	0.4	0.1	**0.9**	...

Feature Extraction

Transforming text data into numeric features

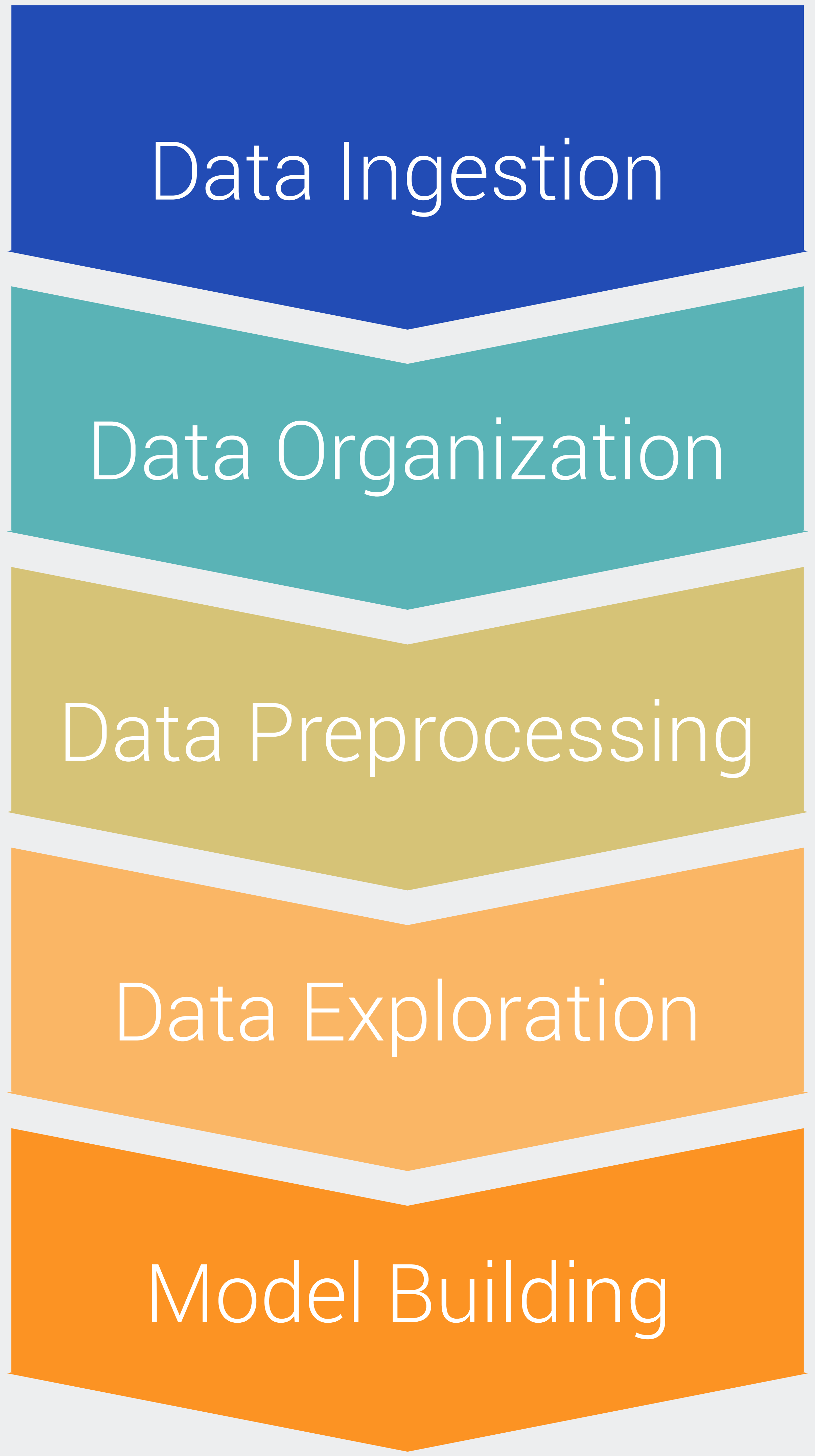
Non-distributed			Distributed
One Hot Encoding	Bag of Words	TF-IDF	Embeddings
Stop word removal?	Stop word removal	No need for stop word removal	Stop word removal?
None	Document-level normalization	Corpus and document-level normalization	Context and corpus-level normalization
All tokens are equidistant			Distance $\propto$ token similarity
High dimensionality, extremely sparse			Lower dimensionality

Document categorization

Extracting semantic structure from numeric features

Topic Modeling	Document Classification
What are the topics that occur in a collection of documents?	Which class does document X belong to?
Unsupervised Dimension Reduction (LDA / LSA)	Supervised ML Algorithms Regex (Standard or Artisanal)
Every document is a mixture of topics	Every document belongs to a single class
Every topic is a mixture of words	The presence or absence of a subset of words impacts the classification

Further Reading



Applied Text Analysis with Python by Benjamin Bengfort, Rebecca Bilbro & Tony Ojeda

Natural Language Processing with Python by Steven Bird, Ewan Klein & Edward Loper

Speech and Language Processing by Dan Jurafsky & James Martin

Foundations of Statistical Natural Language Processing by Chris Manning & Hinrich Schutze

Text Mining with R by Julia Silge & David Robinson

Thank you R Ladies!

Maryam Jahanshahi Ph.D.
@mjahanshahi
in maryam-j

bit.ly/rladies2019