



# PySpark and NLP

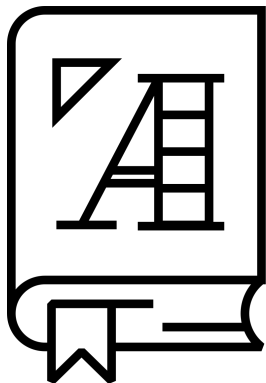
Data Boot Camp  
Lesson 22.2



# Natural Language Processing

# What Is Natural Language Processing (NLP)?

---



A field of study that combines linguistics and computer science in order for computers to interact with human natural languages.

# What Is NLP Used For?

---



Spell checkers



Virtual assistants (Alexa, Google Home, Siri)



Spam filters



Virtual translations (Google Translate)



Handling unstructured data from Tweets and Facebook posts

# Big Data + NLP

---

Most industries have large quantities of textual data that can't be efficiently processed manually.

01

## **Law:**

Research, notes,  
documents, records  
of legal  
transactions,  
governmental  
information

02

## **Medical Research:**

Patient  
information/history,  
clinical notes,  
symptoms

03

## **Stock Market Analysis:**

Company  
disclosures, news  
articles, report  
narratives

# A Few NLP Applications

---



**Text Classification:** Classifying statements as subjective/objective, positive/negative; finding the reading level or genre of a text



**Information Extraction:** Finding the diagnosis from a doctor's notes; identifying names of individuals from a witness statement



**Document Summarization:** Generating a headline or abstract for a document



**Complex Question Answering:** Answering a question about a subject given resources or a document on that subject

# ...But NLP is HARD!

---

Humans intuitively interpret natural language, but even we aren't great at it all the time. Natural language is:



**Contextual:** The meaning of text depends on situation, speaker, and listener.



**Ambiguous:** Words have multiple meanings and can mean different things in different contexts.



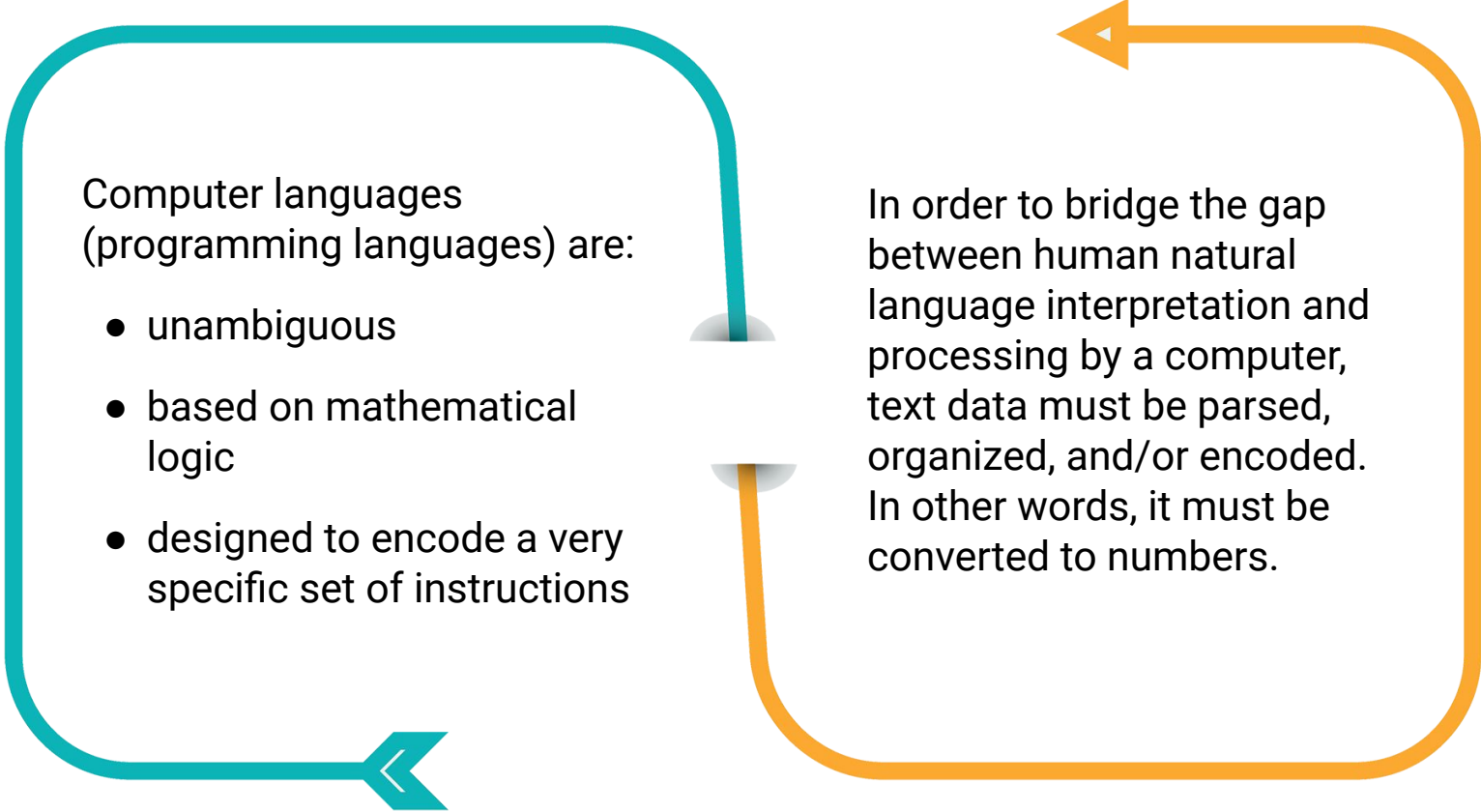
**Nonstandard:** There is no general set of rules, especially across dialects, groups, etc.



And more!

# Natural Languages vs. Computer Languages

---



Computer languages  
(programming languages) are:

- unambiguous
- based on mathematical logic
- designed to encode a very specific set of instructions

In order to bridge the gap between human natural language interpretation and processing by a computer, text data must be parsed, organized, and/or encoded. In other words, it must be converted to numbers.



# Utilizing an NLP Pipeline

---

## Basic NLP Data Pipeline:



- NLP is complicated, so it's helpful to break the process down into smaller steps.
- Each module will complete a separate task.
- The output data from one step then becomes the input data for the next.
- High-level tasks build on low-level tasks.
- This allows evaluation and refinement of each task as needed.
- This is similar to an ML dataset that may require preprocessing, such as encoding or scaling before fitting to a model. However, scaling and encoding can sometimes be done independently.

# Tokenization

# Tokenization

The process of segmenting running text into words, sentences, or phrases.



Text needs to be segmented into units in order for any processing to be done.



A token is a group of characters that have meaning. It can be words, sentences, or phrases.



Sometimes characters such as punctuation are discarded.



Tokenization is similar to using `.split()` in Python.



Sentence segmentation and tokenization are often the first steps in an NLP pipeline.

Let's eat, Grandpa!



["let's", "eat", "grandpa"]



# Stop Words Removal

# Stop Words

---

Stop words are words that are useful for grammar and syntax, but they don't contain any important content.



Generally, stop words are the most commonly used words in the document.



Examples: *this, to, the, a, there, an*



Stop words are often removed because they don't distinguish between relevant and irrelevant content.

# Stop Words in the Pipeline

The quick brown fox  
jumped over the lazy dog.

Tokenize

Tokens

The

quick

brown

fox

jumped

over

the

lazy

dog

Stop words

Tokens

~~The~~

quick

brown

fox

jumped

over

~~the~~

lazy

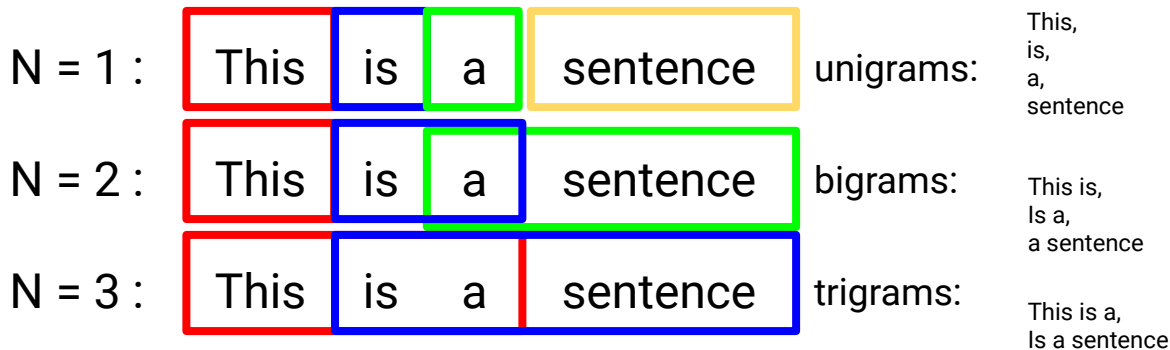
dog

# N-Grams

# N-Grams

A group of  $n$  words appearing in sequence from a text.

- Splitting on single words can result in a model where syntax and order are ignored.
- Using an **n-gram** can be helpful in identifying the multi-word expressions or phrases.
- N-grams can be used to calculate how often words follow one another and are applied in generating text (Predictive Keyboard).
- N-grams are helpful in applications like sentiment analysis, where the ordering of the words is important to the context.



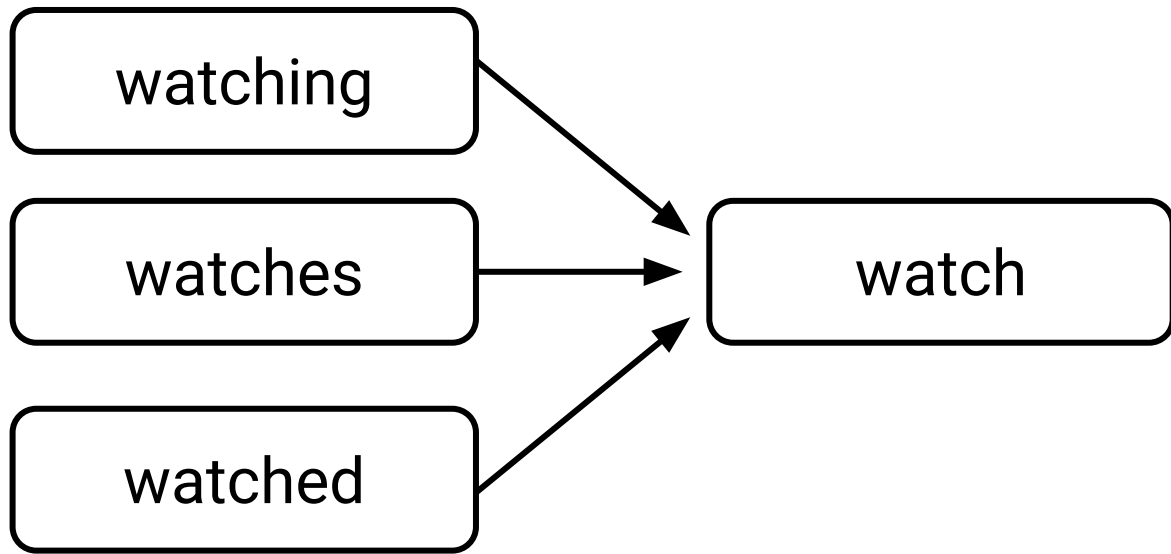


# Stemming

# Stemming

---

The process of stripping a word down to its root word so that different forms of the word are identified as having the same significance in text analysis.



# Term Frequency— Inverse Document Frequency

# TF-IDF

---

A measure intended to reflect the importance of a word in a text.

- TF—Term Frequency: A count of the word in a document
- IDF—Inverse Document Frequency:

$$\log \left( \frac{\text{total number of documents}}{\text{number of documents containing target word}} \right)$$

- IDF: The more documents that include the term, the lower the IDF score.
- TF-IDF is the product of the two. TF drives up the score, but IDF will bring it down if the word occurs in all or many documents.

# TF: Bag of Words

## The Bag of Words Representation

The methods used for term frequency use a “bag of words” approach.

Each document is represented by a “bag of words” where grammar and word order are disregarded, but frequency is kept.

I love this movie! It's sweet but with satirical humor. The dialogue is great and the adventure scenes are fun...It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



It	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1

# Finding TF: CountVectorizer

---

CountVectorizer is used to convert a collection of text to vectors of token counts.

Three parameters:



**minTF:** Sets the minimum term frequency of a word. A value of 1 will ensure all words are included.



**minDF:** Sets the minimum document frequency of a word. A value of 1 will ensure all words are included.



**vocabSize:** Limits the size to the top number of words specified. A number greater than or equal to the number of unique words in the corpus will include all words.



The results will be returned in descending values and based on the term's frequency across the entire corpus, not just the document, and will eliminate words based on parameters.

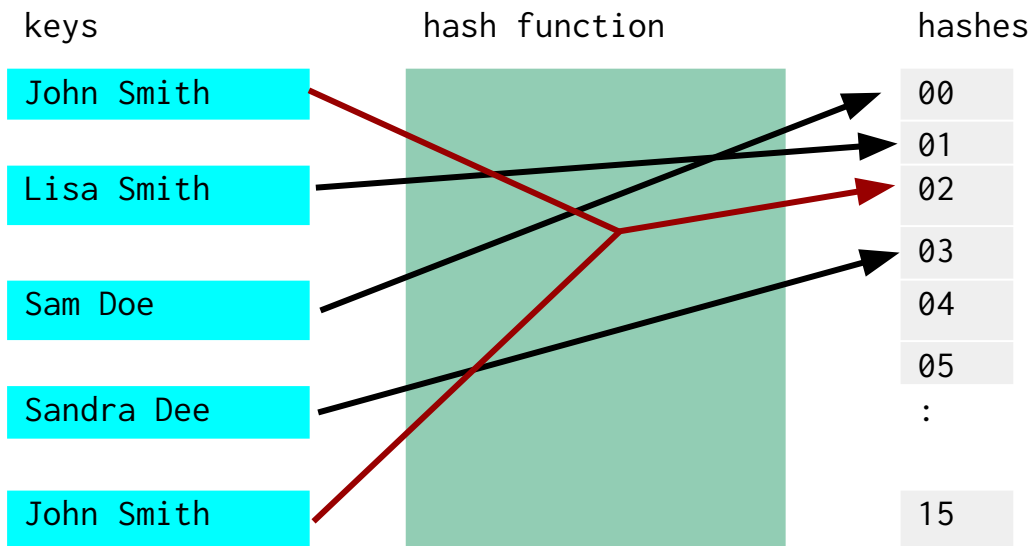
# Finding TF: HashingTF



Hashing is the process of transforming data of arbitrary size to a fixed size.

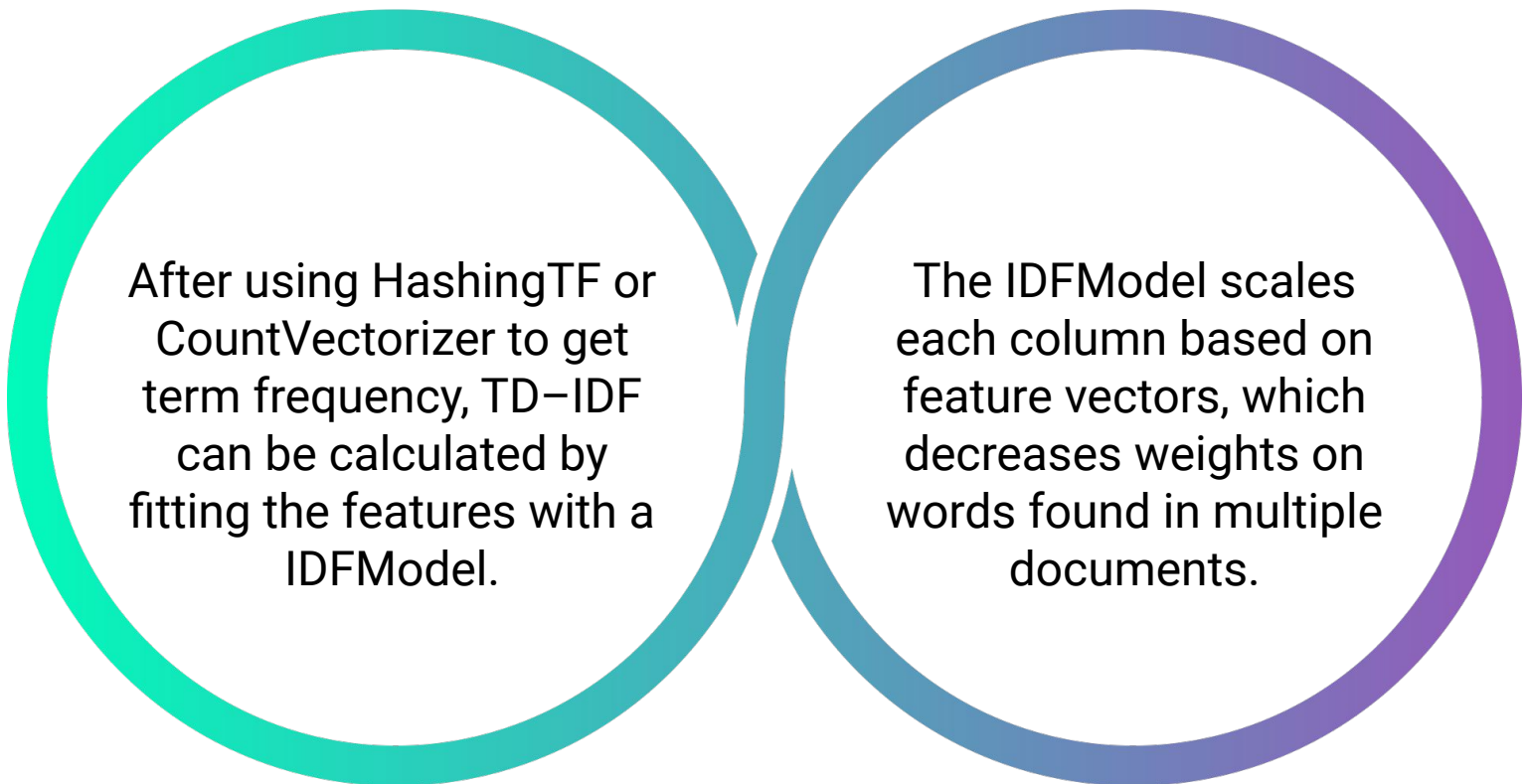


Each word is input in a hash function to give it a number. The same words result in the same hash ID.



# The IDFModel

---



After using HashingTF or CountVectorizer to get term frequency, TD-IDF can be calculated by fitting the features with a IDFModel.

The IDFModel scales each column based on feature vectors, which decreases weights on words found in multiple documents.