Preparing Text Data for Machine Learning

Janani Ravi
CO-FOUNDER, LOONYCORN
www.loonycorn.com

Overview

Representing text in numeric form

One-hot, frequency-based and prediction-based embeddings

Bag-of-words and bag-of-ngrams modeling of text

Building feature vectors from text data using CountVectorizer, TfIdfVectorizer and HashingVectorizer

Performing feature extraction on a Python dictionary

Encoding Text Data in Numeric Form

```
d = “This is not the worst restaurant in the metropolis,
not by a long way”
```

Document as Word Sequence

Model a document as an ordered sequence of words

```
d = “This is not the worst restaurant in the metropolis,
not by a long way”
```

```
(“This”, “is”,”not”,”the”,”worst”,”restaurant”,”in”,”the”,
“metropolis”, “not”,”by”,”a”,”long”,”way”)
```

Document as Word Sequence

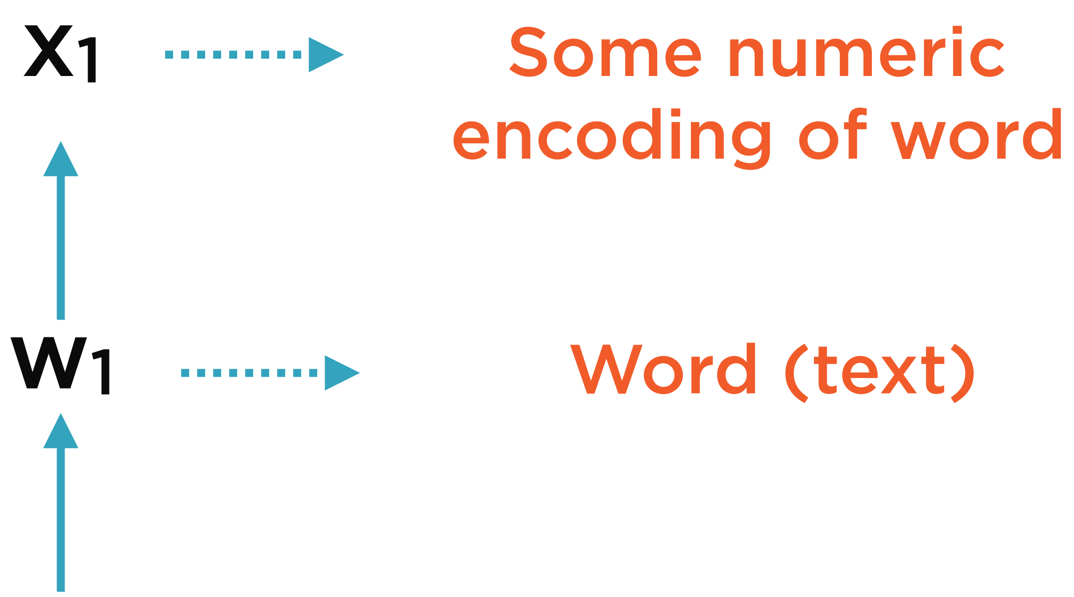
Tokenize document into individual words

x_0 → Some numeric encoding of word

w_0 → Word (text)

("This", "is","not","the","worst","restaurant","in","the", "metropolis", "not","by","a","long","way")

Represent Each Word as a Number



(“This”, “is”,”not”,”the”,”worst”,”restaurant”,”in”,”the”, “metropolis”, “not”,”by”,”a”,”long”,”way”)

Represent Each Word as a Number

x_n Some numeric encoding of word

w_n Word (text)

```
("This", "is","not","the","worst","restaurant","in","the",
"metropolis", "not","by","a","long","way")
```

Represent Each Word as a Number

$$d = [x_0,\ x_1,\ \dots\ x_n]$$

Document as Tensor

Represent each word as numeric data, aggregate into tensor

Numeric Representations of Text

One-hot

Frequency-based

Prediction-based

Numeric Representations of Text

One-hot

Frequency-based

Prediction-based

Represent each word in text by its presence or absence

Numeric Representations of Text

One-hot

Frequency-based

Prediction-based

Frequency-based Embeddings

Count

TF-IDF

Co-occurrence

Frequency-based Embeddings

Count

TF-IDF

Co-occurrence

Capture how often a word occurs in a document i.e. the **counts** or the **frequency**

Frequency-based Embeddings

Count

TF-IDF

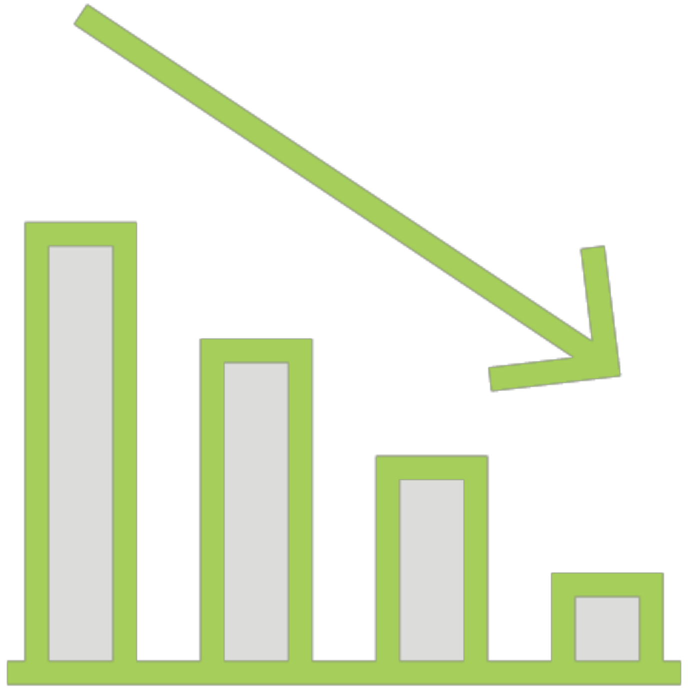
Co-occurrence

Captures how often a word occurs in a **document** as well as the **entire corpus**

Tf-Idf

Frequently in a single document

Might be important



Frequently in the corpus

Probably a common word like "a", "an", "the"

Frequency-based Embeddings

Count

TF-IDF

Co-occurrence

Similar words will occur together and will have similar context

Context Window

A window centered around a word, which includes a certain number of neighboring words

Co-occurrence

The number of times two words *w1* and *w2* have occurred together in a context window

Word Embeddings

One-hot

Frequency-based

Prediction-based

Predictions-based embeddings

Numerical representations of text which capture meanings and semantic relationships, generated using ML models

Magic

Word embeddings capture meaning

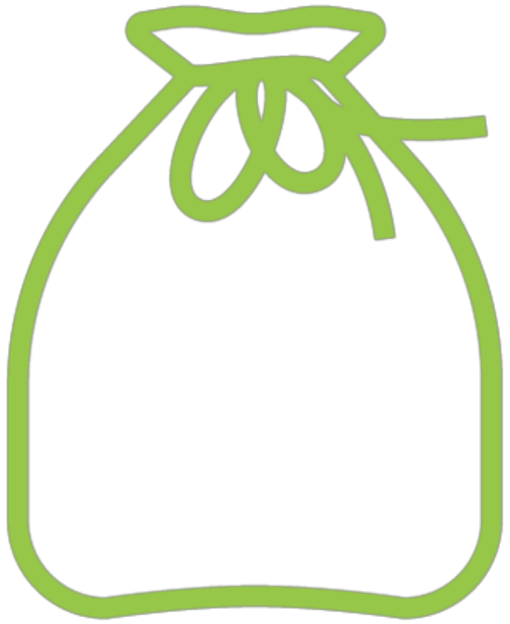
“Queen” ~ “King” == “Woman” ~ “Man”

“Paris” ~ “France” == “London” ~ “England”

Dramatic dimensionality reduction

Bag-based Models for Text

Bag-based Models for Text



Bag-of-words

Bag of n-grams

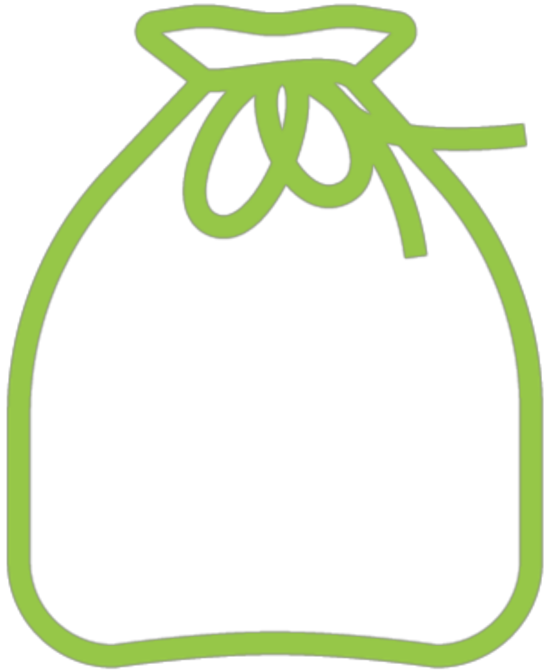
Bag-of-words Model

Any model that represents the document as a bag (multiset) of its constituent words, disregarding order but maintaining multiplicity

Bag-of-words Model

Any model that represents the document as a bag (multiset) of its constituent words, disregarding order but maintaining multiplicity

Bag-of-words Models



Examples of bag-of-words models

- Count Vectorization
- TF-IDF Vectorization

Examples that are not bag-of-words models

- One-hot encoding (no multiplicity)
- Word embeddings

Bag-of-n-grams Model

Any model that represents the document as a bag (multiset) of its constituent n-grams, disregarding order but maintaining multiplicity

Bag-of-n-grams Model

Any model that represents the document as a bag (multiset) of its constituent n-grams, disregarding order but maintaining multiplicity

Bag-of-n-grams

Bag is a set with duplicates (i.e. multiset)

Bag-of-words models contain only individual words

Bag-of-n-gram models contain n-grams

Bag-of-n-grams

An n-gram model store additional spatial information for a word

Words that occur together

Demo

Vectorize text data using the bag-of-words model

Demo

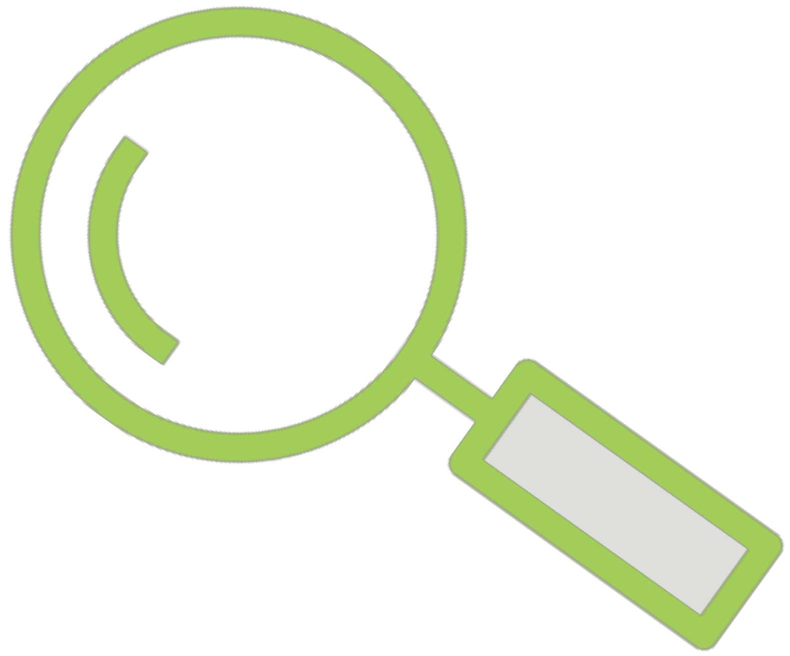
Vectorize text data using the bag-of-n-grams model

Demo

Vectorize text data using tf-idf scores

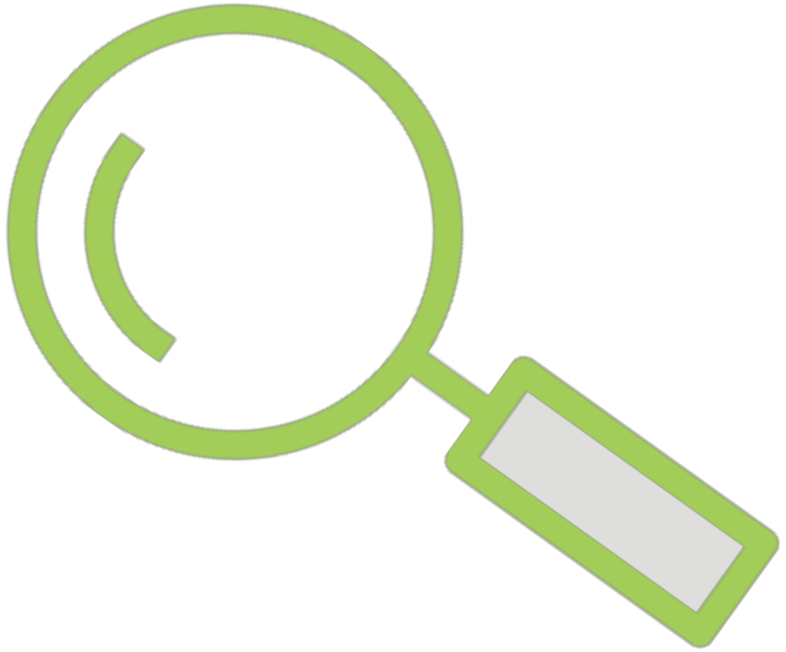
Hashing

Hashing



A technique that allows you to lookup specific values very quickly

Hashing



Also can be used to perform dimensionality reduction

Hashing

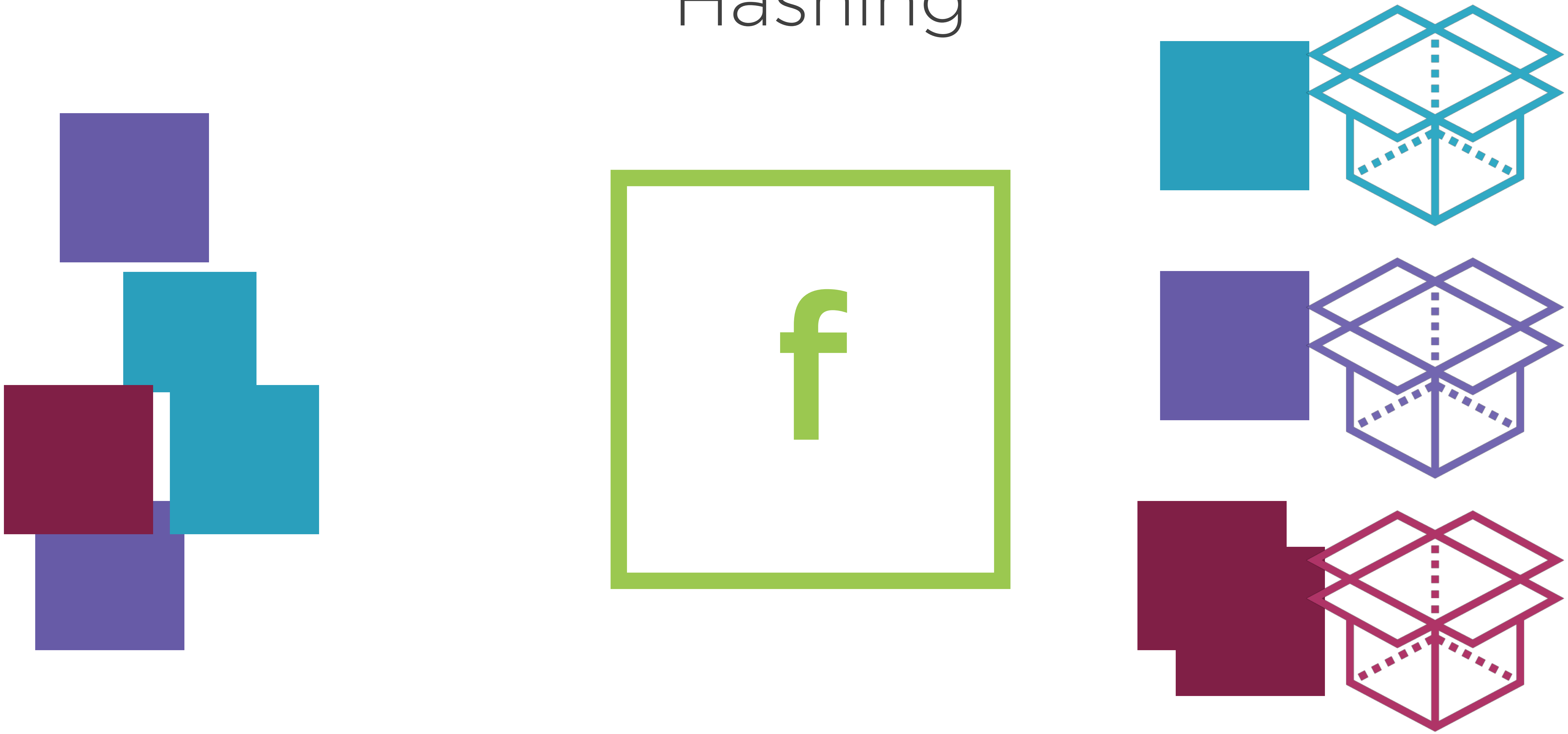
Have a fixed number of categories or buckets

Hashing

A hash function determines which bucket each value belongs to

Hashing

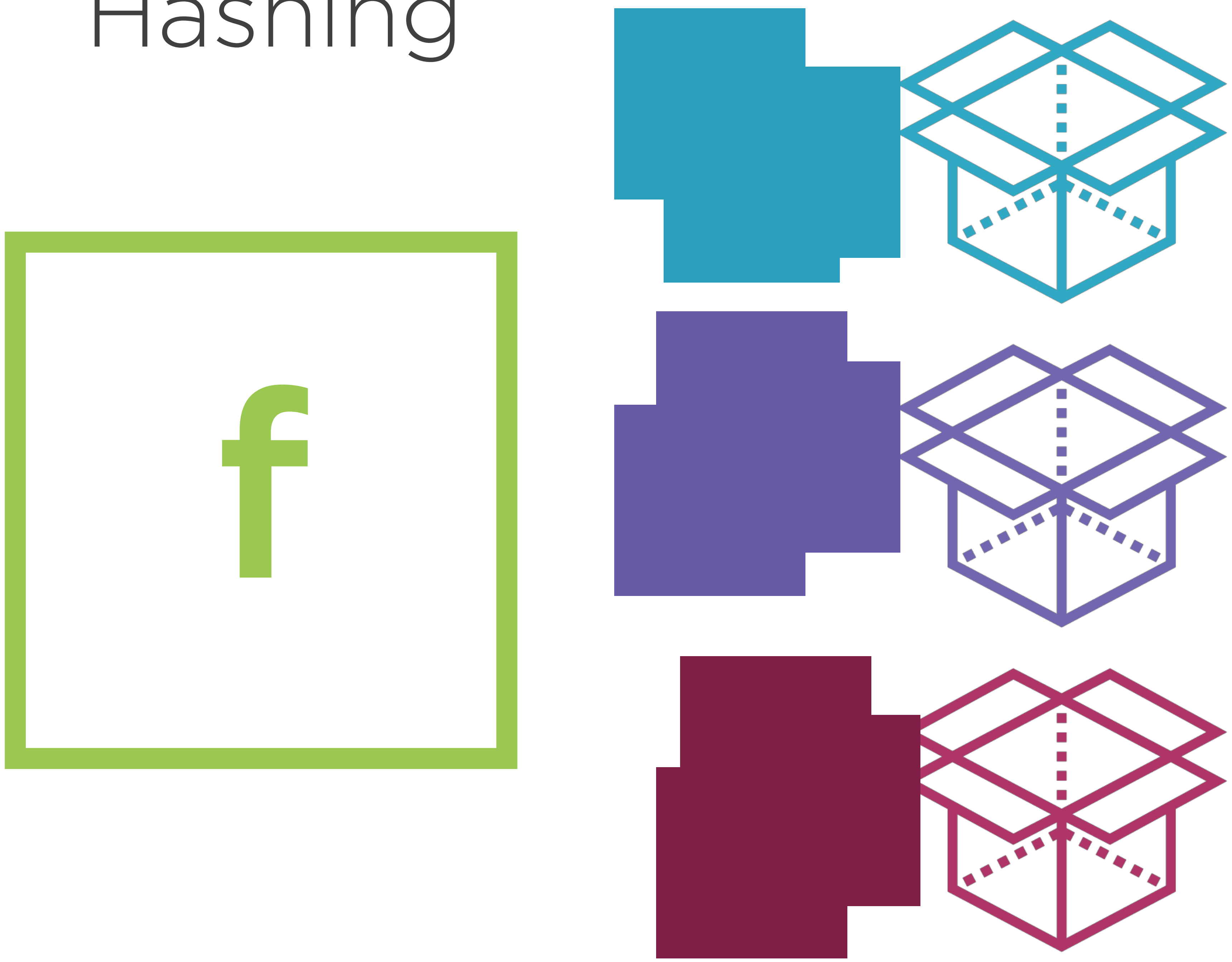
Hashing



Hashing

Hashing

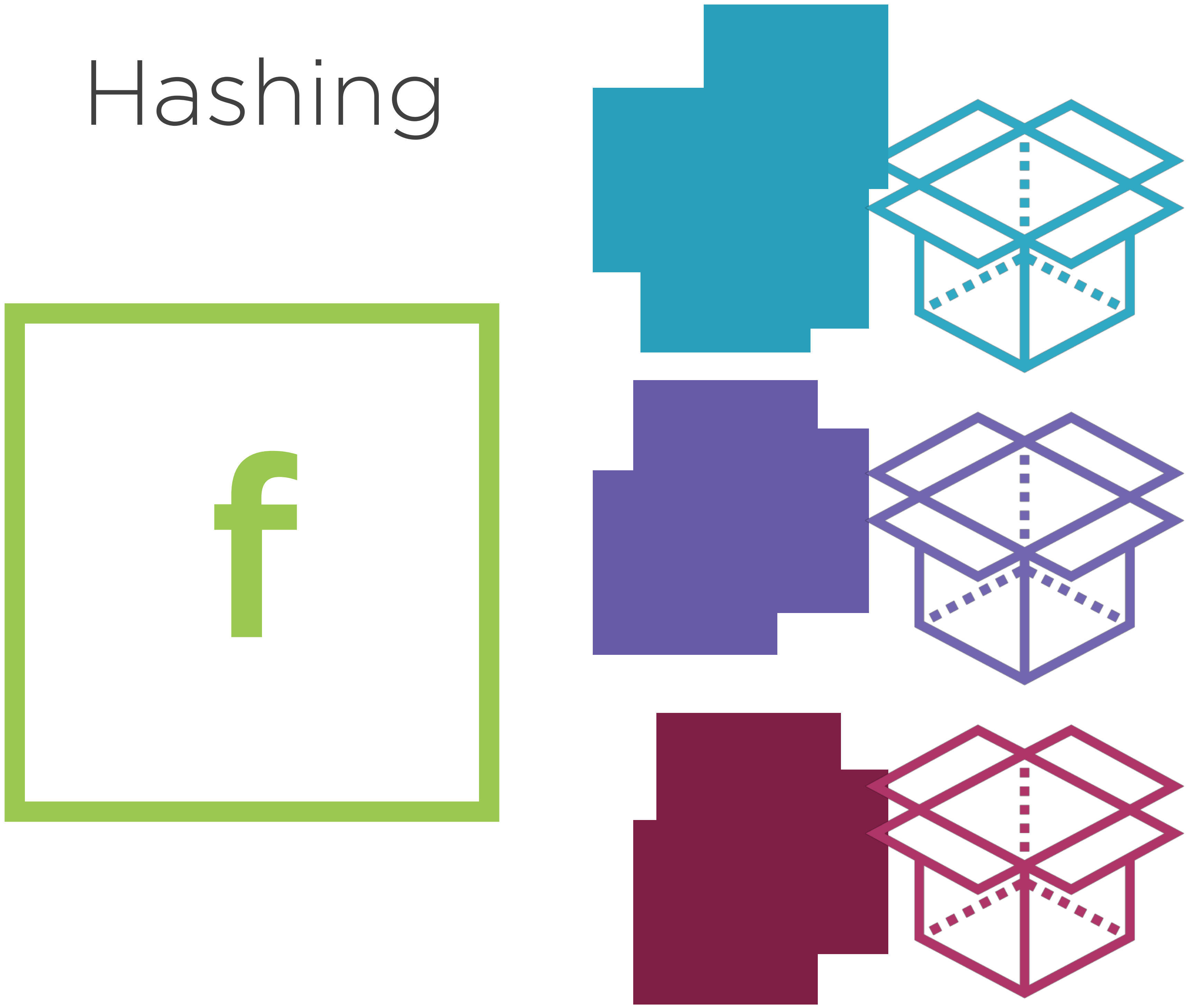
Hashing



Hashing

For any new value we know immediately which bucket it belongs to

Hashing



For any new value we know immediately which bucket it belongs to

Hashing

Each value is hashed so it falls in one of these buckets

Hashing

A value can only belong to **one bucket** and always belongs to the **same bucket**

Feature Hashing in Text

Apply a hash function to words to determine their location in the feature vector representing a document. Fast and memory efficient but has no inverse transform.

Feature hashing uses the "**hashing trick**" for dimensionality reduction

Dimensionality Reduction

Input: N-dimensional data

Output: k-dimensional data

Where k < N

Hashing

Input: N-dimensional data

Output: 1-dimensional data

Output is the hash bucket the data maps to

Hashing

Input: N-dimensional data

Output: k-dimensional data

Can easily extend hashing to output desired dimensionality

Hashing

Thus hashing is a simple form of dimensionality reduction

However, very similar inputs may be mapped to very different hash values

Hashing performs dimensionality reduction but does not keep similar data points together

Demo

Reducing dimensions in text using the hashing vectorizer

Demo

Performing feature extraction on a Python dictionary

Summary

Representing text in numeric form

One-hot, frequency-based and prediction-based embeddings

Bag-of-words and bag-of-ngrams modeling of text

Building feature vectors from text data using CountVectorizer, TfIdfVectorizer and HashingVectorizer

Performing feature extraction on a Python dictionary