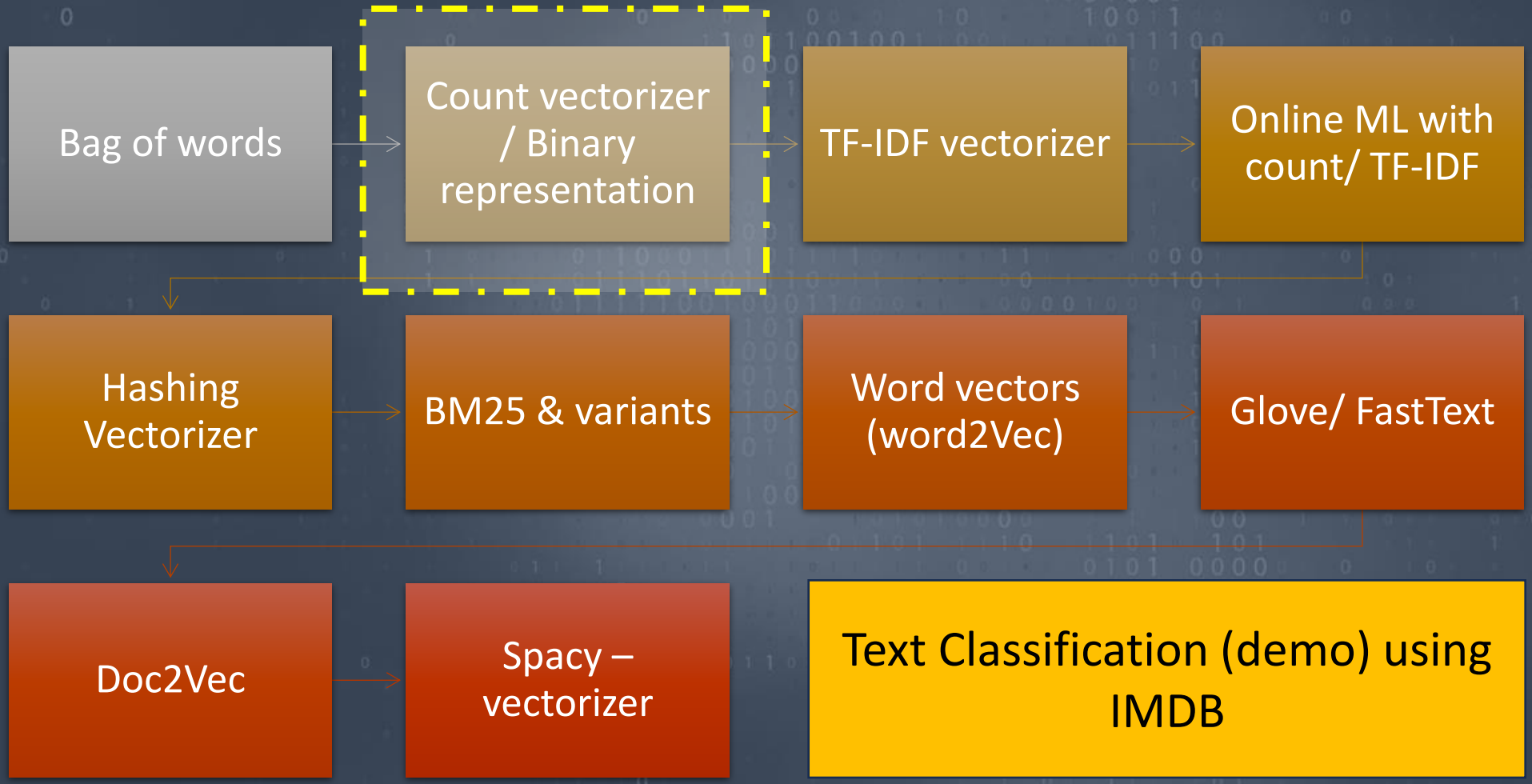
Vectorization & Embedding

"Textual Alchemy: Words to numbers"

GROKKERS
AI FOR EVERYONE

Topics to cover



Count Vectorizer

Frequency, Binary

CountVectorizer

is a text preprocessing technique commonly used in natural language processing (NLP) to convert a collection of text documents to a matrix of token counts.

It's a part of the bag-of-words model, which represents text data as a set of individual words without considering the order or structure.

CountVectorizer is often used as a feature extraction method for machine learning models.

CountVectorizer in scikit-learn

Input

- input (default='content'): It can be 'content' for a sequence of strings or 'filename' for a sequence of filenames.

Lowercase

- lowercase (default=True): If True, it converts all characters to lowercase before tokenizing.

Token

- token_pattern (default='(?u)\b\w\w+\b'): Regular expression for tokenizing. The default token pattern extracts words of two or more alphanumeric characters.

Stop

- stop_words (default=None): Specifies a list of stop words that will be removed from the vocabulary.

… some more parameters…

ngram_range (default=(1, 1)):

- Specifies the range of n-grams to consider. For example, (1, 1) means unigrams, (1, 2) means unigrams and bigrams, and (2, 2) means only bigrams.

analyzer (default='word'):

- Determines whether to analyze words or characters. It can be 'word' or 'char'.

max_df (default=1.0):

- Ignores terms that have a document frequency strictly higher than the given threshold (float or integer).

min_df (default=1):

- Ignores terms that have a document frequency strictly lower than the given threshold (float or integer).

max_features (default=None):

- Limits the number of features. If None, all features are used.

Use case of count vectorizer

Text Classification:	• **Example:** Spam detection, sentiment analysis, topic categorization.
Document Clustering:	• **Example:** Organizing news articles into clusters of related topics.
Information Retrieval:	• **Example:** Search engines returning relevant web pages based on search queries.
Feature Engineering for Machine Learning:	• **Example:** Predicting user preferences based on product reviews.
Keyword Extraction:	• **Example:** Summarizing articles by extracting key terms.

Limitations - Ignores Word Order and Structure

CountVectorizer treats each document as a bag of words, disregarding the order and structure of words in a sentence.

limitation may lead to a loss of important information, especially in tasks where word order matters, such as sentiment analysis or language modeling.

… more limitations

Fixed Size Vocabulary

- The vocabulary created by CountVectorizer has a fixed size determined by the unique words in the training data.
- New words in test or unseen data won't be represented unless the vocabulary is updated, potentially leading to information loss.

Sensitivity to Stop Words

- Commonly used words (stop words) are often ignored by CountVectorizer based on the stop_words parameter.
- While this can be beneficial for some tasks, it may lead to the exclusion of important context-carrying words.

… more limitations

Equal Importance to All Words:

- CountVectorizer assigns equal importance to all words based on their frequency.
- However, some words may be more informative or carry more significance than others.
- Techniques like TF-IDF (Term Frequency-Inverse Document Frequency) can be used to address this, but they are not inherently part of CountVectorizer.

Sparsity of the Matrix:

- The resulting count matrix can be highly sparse, especially when dealing with a large vocabulary or sparse text data.
- This can lead to increased memory requirements and computational inefficiency.

… some more limitations

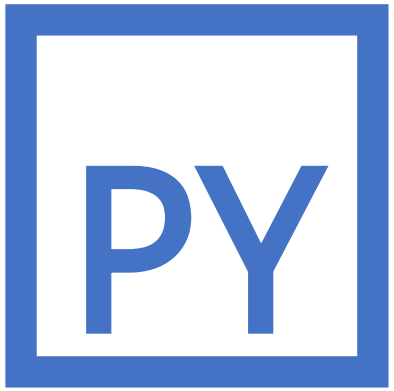
Limited Context Understanding:

- CountVectorizer considers each word independently, without considering the context in which it appears.
- This limitation can be problematic for tasks that require understanding the meaning of phrases or expressions.

Not Suitable for Semantic Analysis:

- CountVectorizer does not capture semantic relationships between words.
- For tasks that require understanding the semantic meaning of words and their relationships, more advanced methods like word embeddings or contextual embeddings (e.g., Word2Vec, GloVe, BERT) are more suitable.

Demo using python/sklearn



(Implement text classification using count vectorizer)

Thanks!