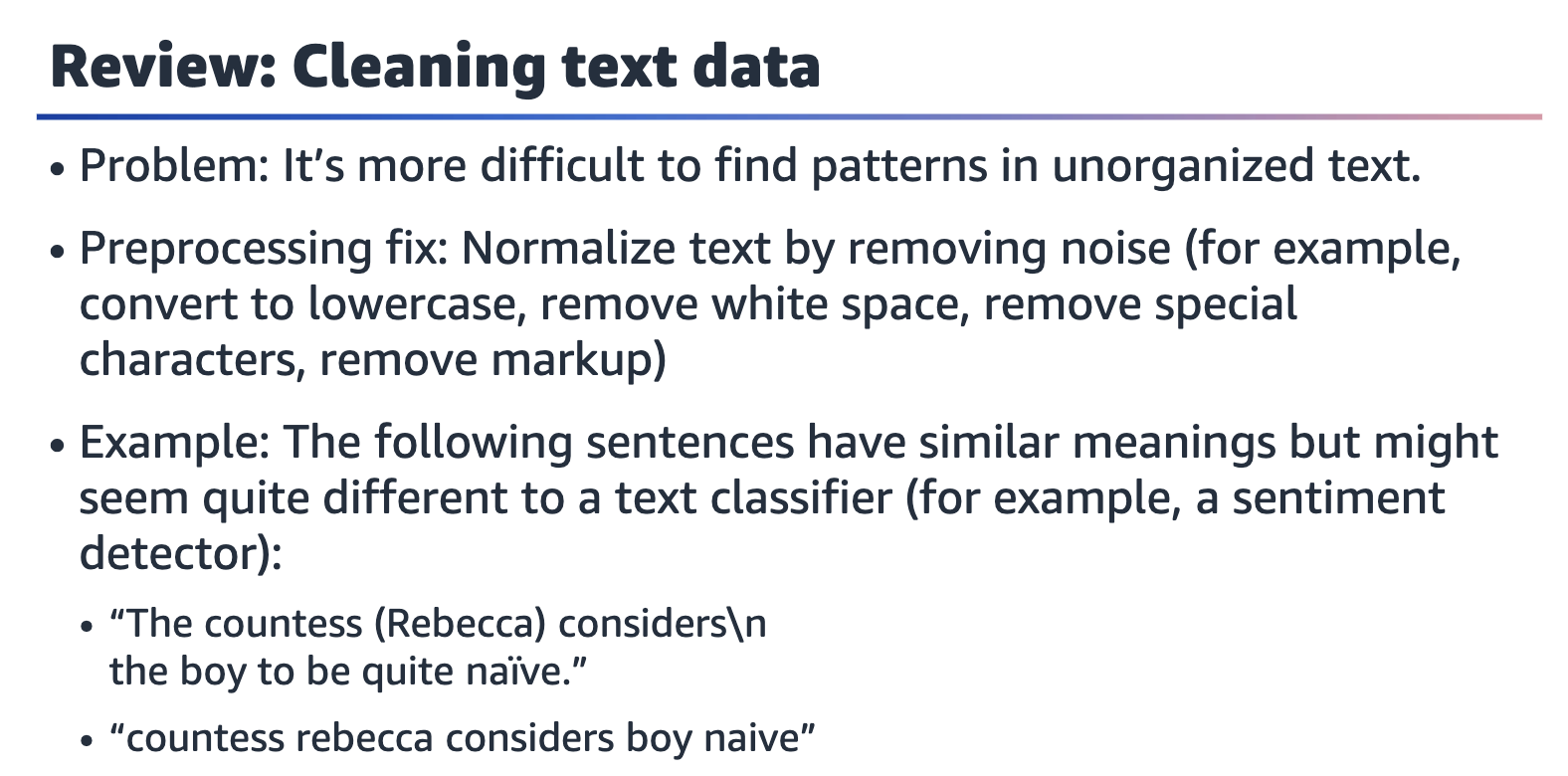


Process flow with four steps. Text data points to text preprocessing, which points to vectorization. Vectorization points to train the ML model. Text preprocessing and vectorization are highlighted with a note that indicates this lesson will explore these elements one at a time.

How can you use text data in ML? Text is text, but ML works with well-defined numerical data. The process involves **taking text data**, **preprocessing** it, **vectorizing** it, and then finally **using it to train your ML model**. This lesson will explore the text preprocessing and vectorization steps.



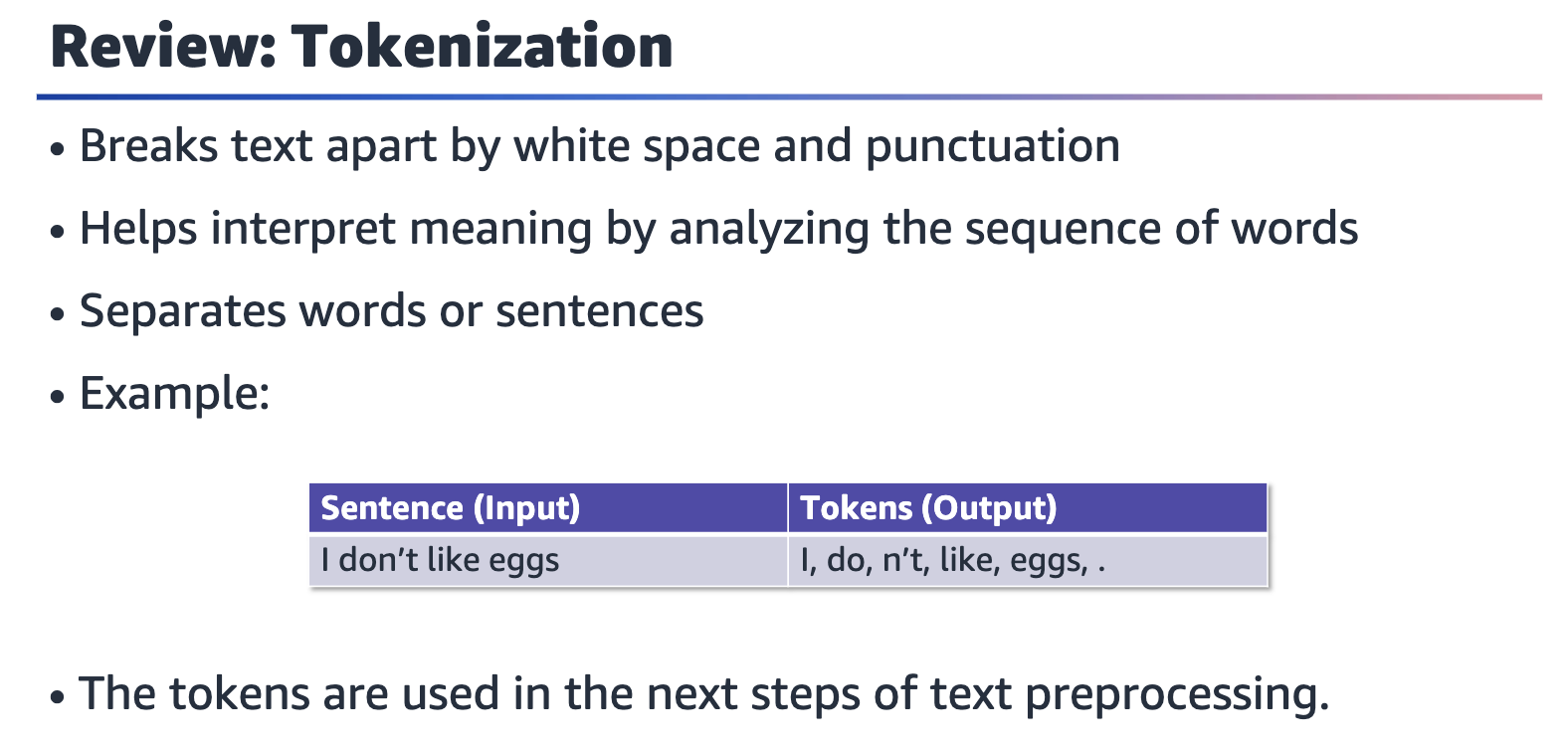
Text preprocessing involves cleaning the raw text data before feature engineering.



Let's review how to clean text data.

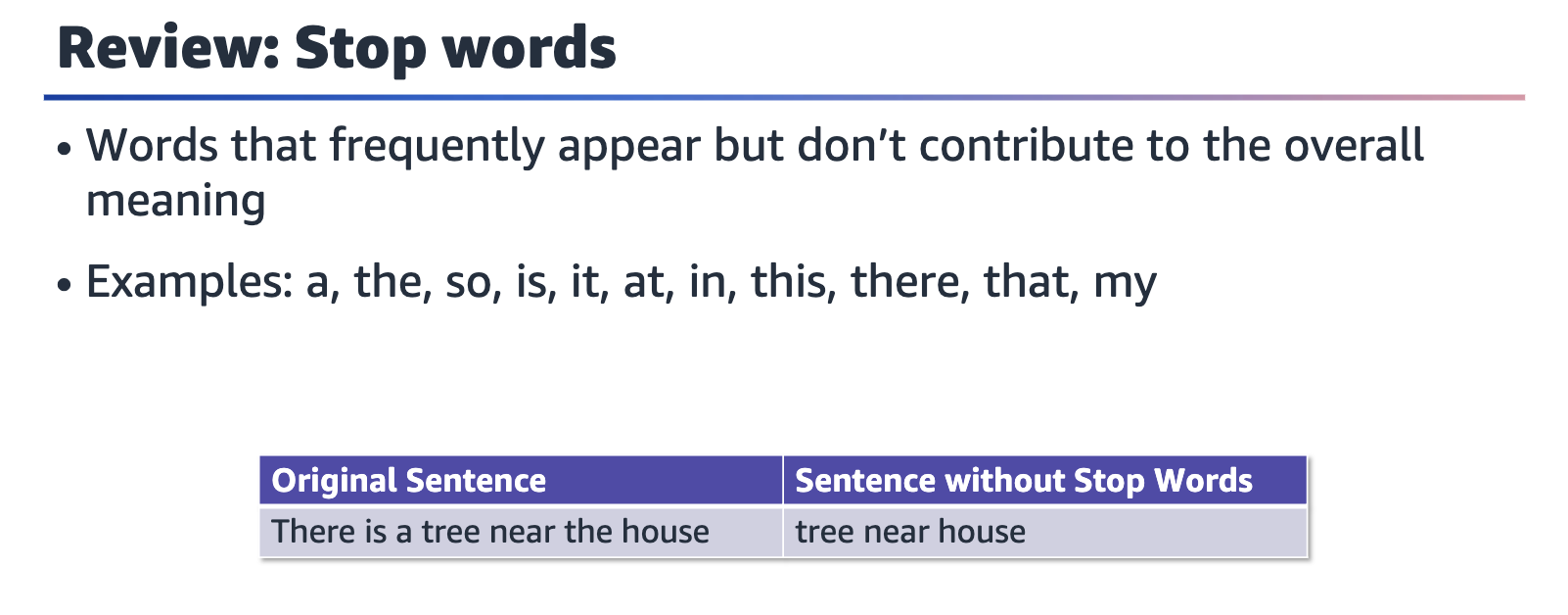
Besides the preprocessing options that are listed on the slide, text processing libraries have quite a bit more to offer.

Simpler text data, simpler models, and smaller vocabularies are better. However, keep in mind that cleaning text data is truly task specific. For example, in sentiment analysis, some words might need to stay although you would otherwise remove them —such as quite in the example sentence on the slide.



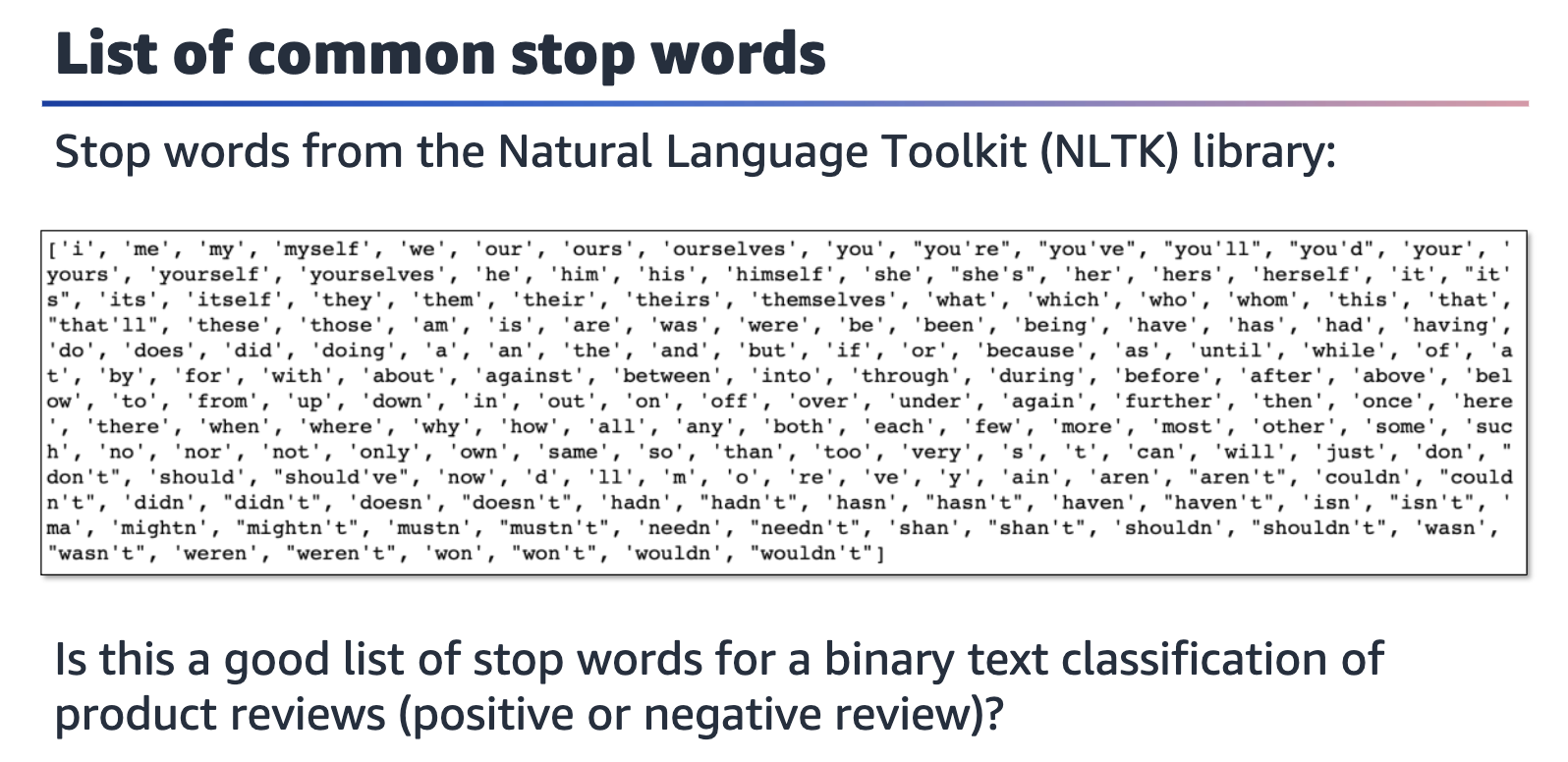
Let's review tokenization.

Tokenization is a key part of text processing. Tokens are the smallest pieces of text data that are extracted from documents. This process produces tokens by splitting the text by white space and punctuation. Note that this is language specific- only English is being discussed here.



Let's review stop words in text data.

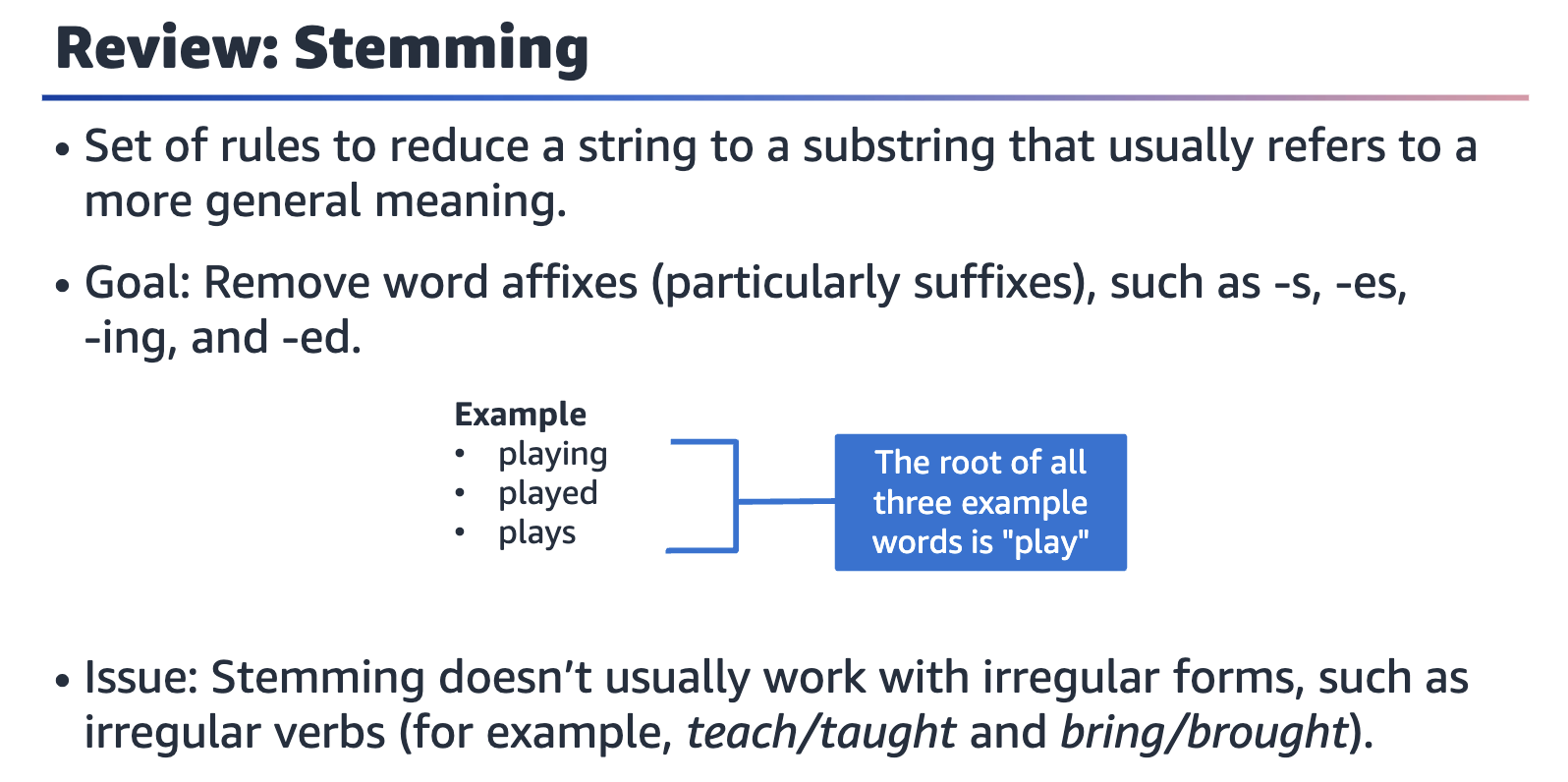
After tokenization, you have tokens. The stop words removal process has a list of stop words, which are frequent words that don't contribute much to the overall meaning of the text. You compare your tokens with the stop word list and remove the tokens that are on it.



The Natural Language Toolkit (NLTK) is a popular Python library to work with and model text. The toolkit has tools to load and clean text. NLTK also provides an extensive list of common stop words for a variety of languages. The image on the slide shows the common stop words for English.

Add stop words as needed or decide to allow some of these words in your text. The decision depends on your particular task. For example, against might be a meaningful word to have in a binary classification task for product reviews (positive or negative review).

You can print a list of stop words from the NLTK library. For more information, see the NLTK documentation at <http://www.nltk.org>.



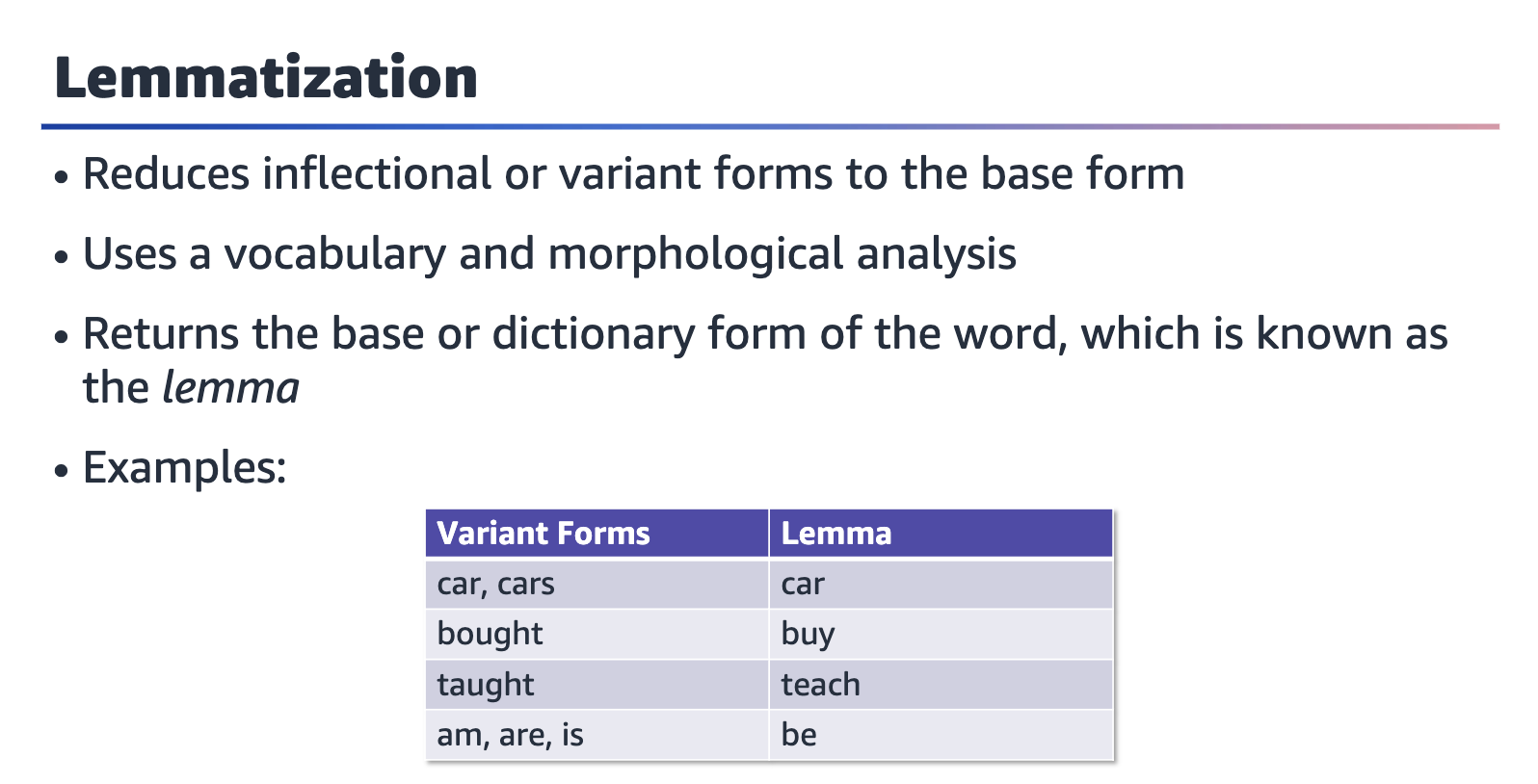
Let's review stemming.

To further reduce the vocabulary and focus on the general meaning of the words, rather than deeper meaning, you can reduce words further by removing affixes— in particular, suffixes. For example, the words *playing*, *played*, and *plays* all reduce to *play* through the process of stemming.

Note that stemming doesn't usually work with irregular forms, such as the irregular verbs *taught* and *brought* in English.

You can use your own text dataset to experiment with stemming rules and other lexicon-based cleaning options, such as lemmatization.

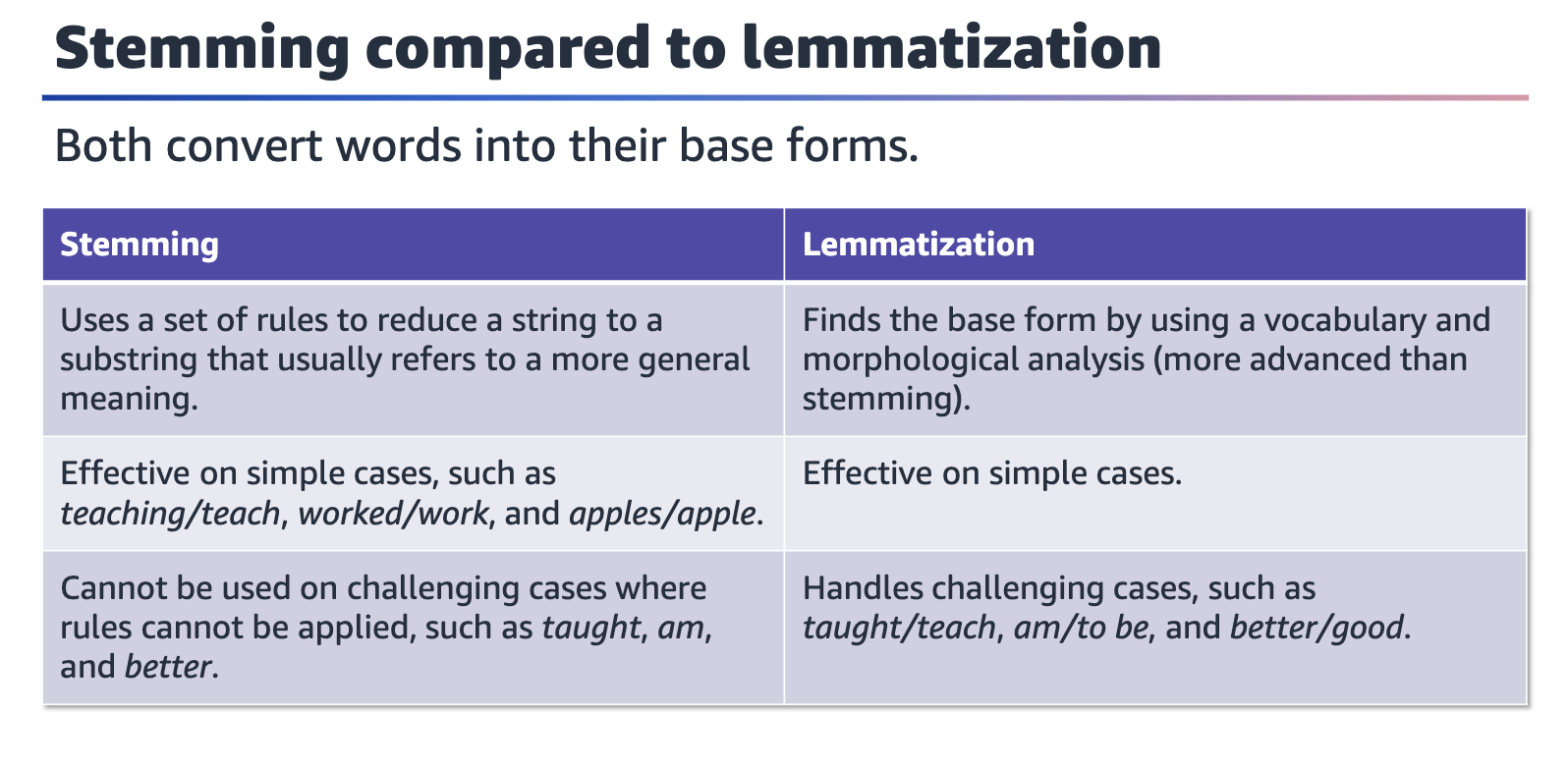
Note that the texts that have been shown so far are small and would be cleaned or processed quickly. Real text data is frequently long and messy-text preprocessing might take a while to complete.



Lemmatization is an important aspect of natural language understanding (NLU) and natural language processing (NLP). Similar to stemming, lemmatization reduces inflection forms and sometimes derivationally related forms of a word to a common base word.

Stemming relies on a simple heuristic process to remove derivational affixes, but lemmatization transforms text by using a vocabulary and morphological analysis of the words. The objective is to obtain the base or dictionary form of a word, which is known as the lemma.

A lemmatizer performs full morphological analysis to accurately identify the lemma for each word. Doing a full morphological analysis produces modest benefits for retrieval.



This slide compares stemming and lemmatization. Both techniques convert words into their base forms, but they do so in different ways.

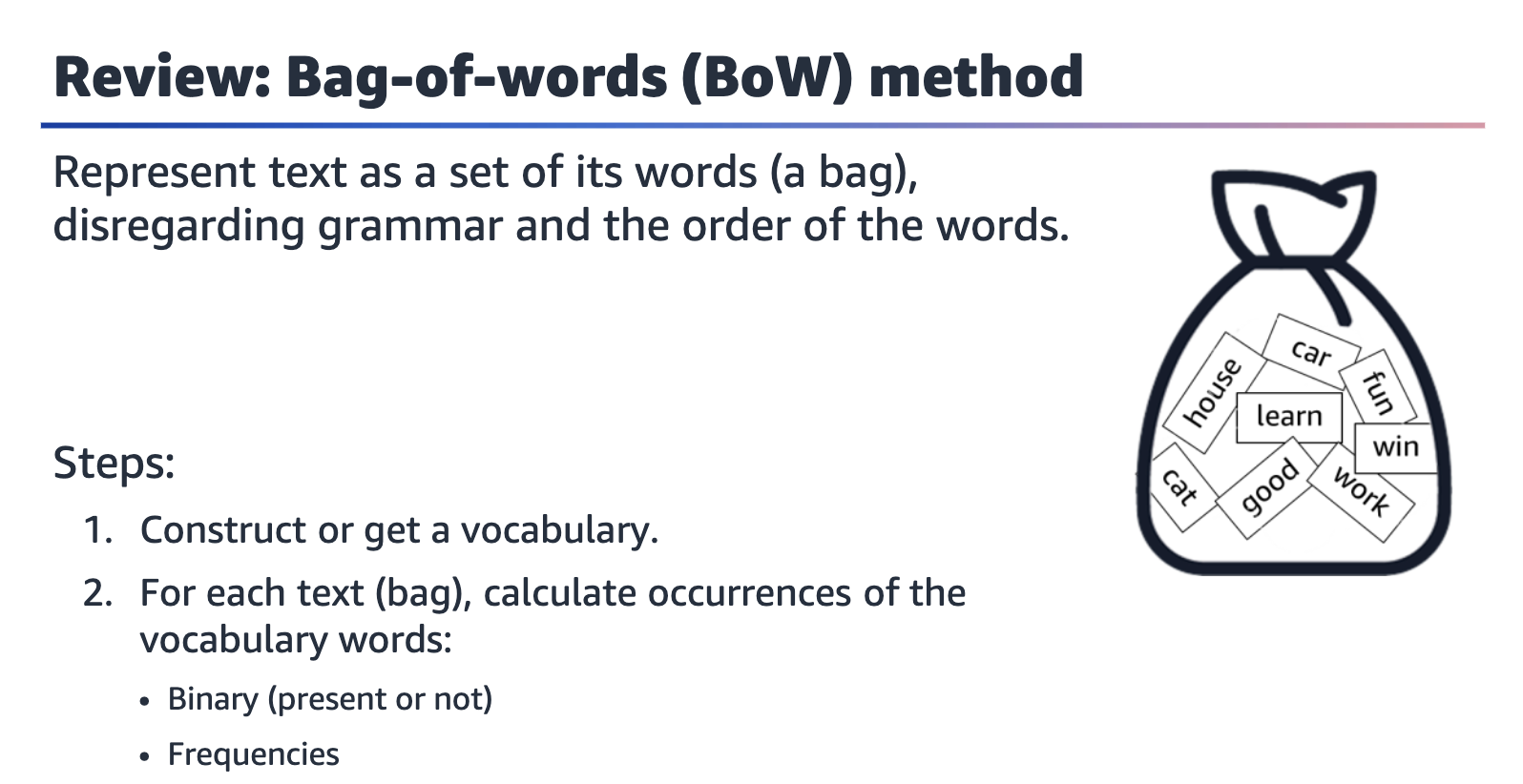
Stemming is a rule-based method that creates substrings that correspond to a simpler form of a given token. The process removes the extras from the start or end of the token.

Lemmatization is more involved. It uses a grammatical approach, so it can handle more difficult cases, such as irregular forms.

The goal of both methods is to produce simpler or more general meaning tokens.



Vectorization involves transforming text features into numerical format.

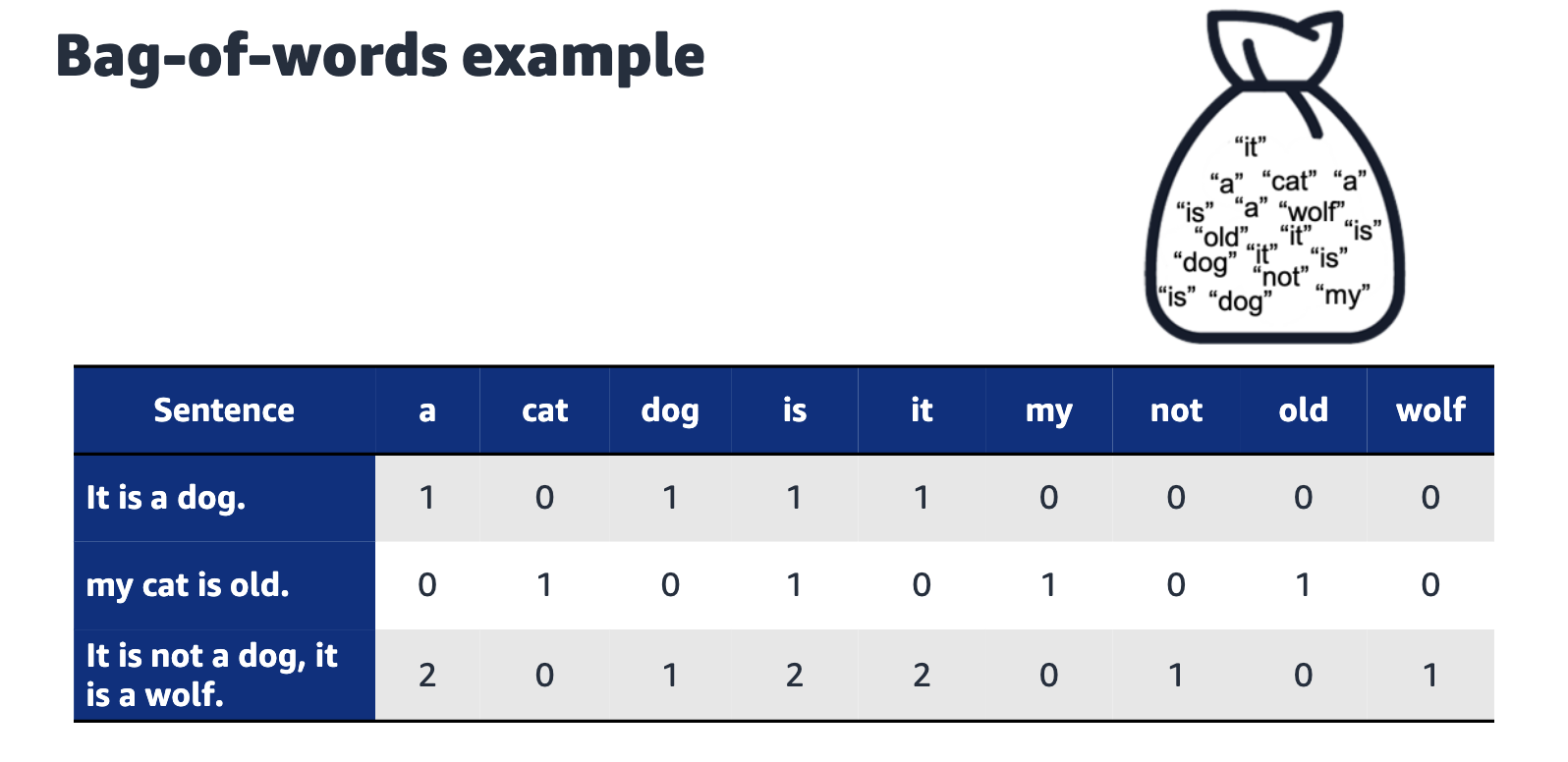


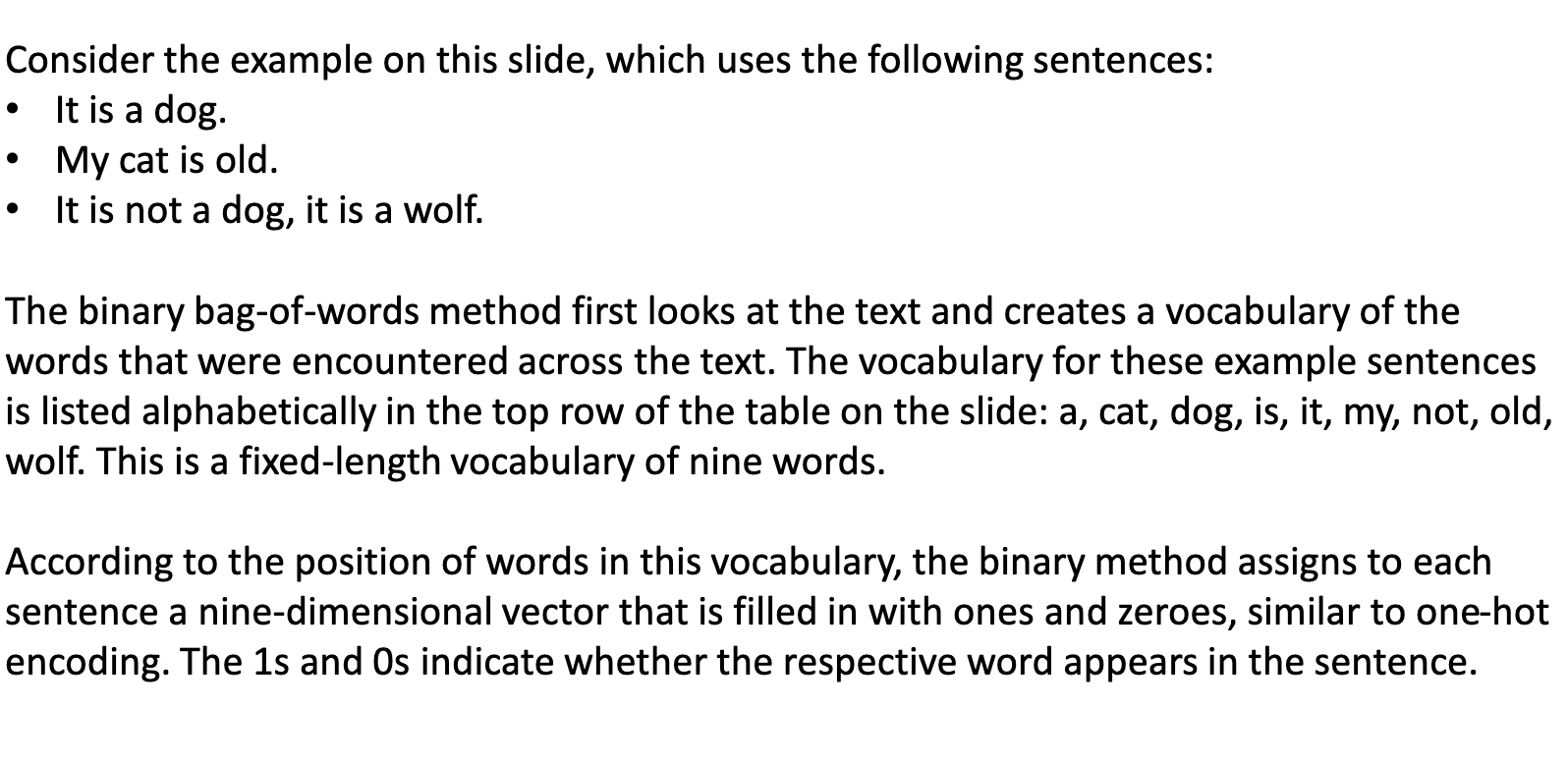
Let's review the bag-of-words (BoW) method to vectorize text.

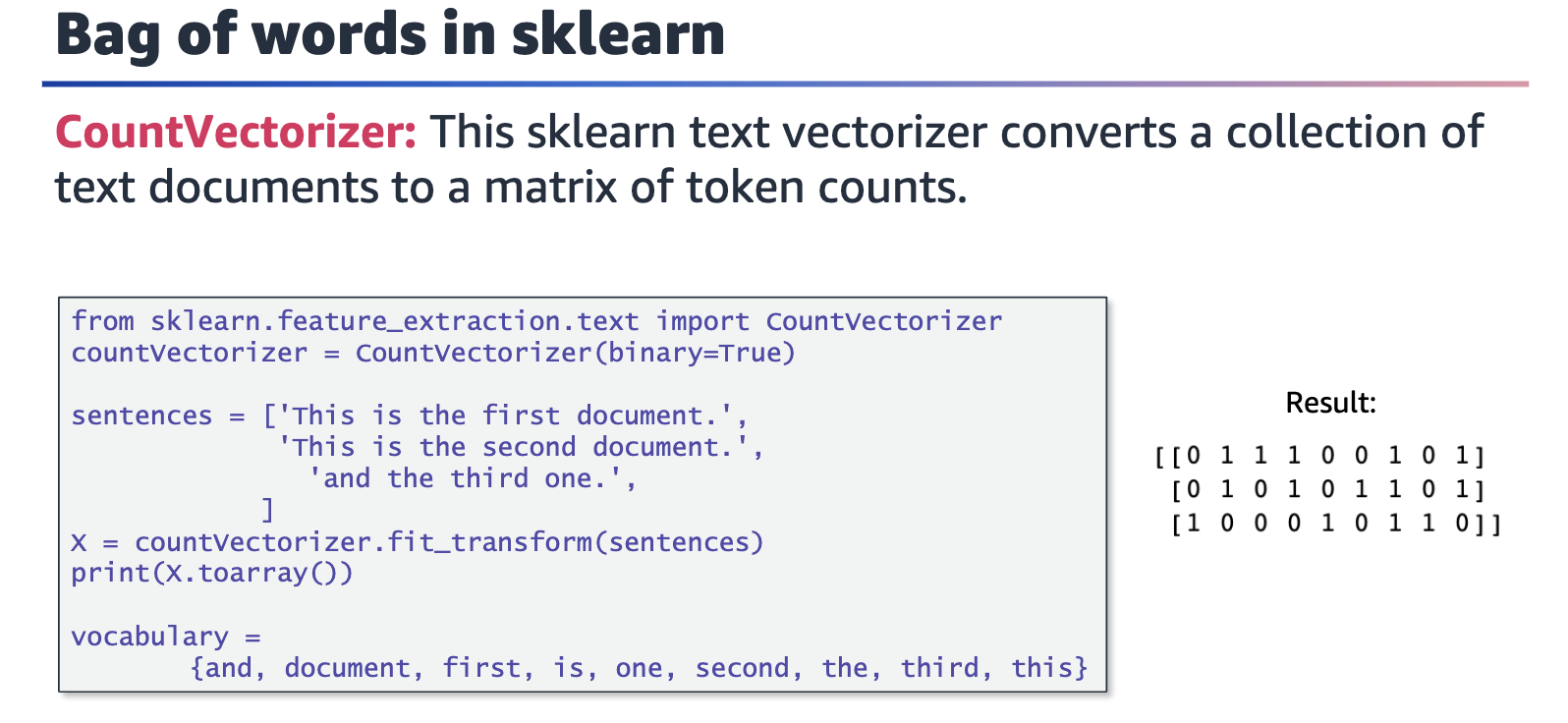
The BoW method is a special technique to convert text data into numerical form. This method is usually considered less modern, but it's still useful to learn.

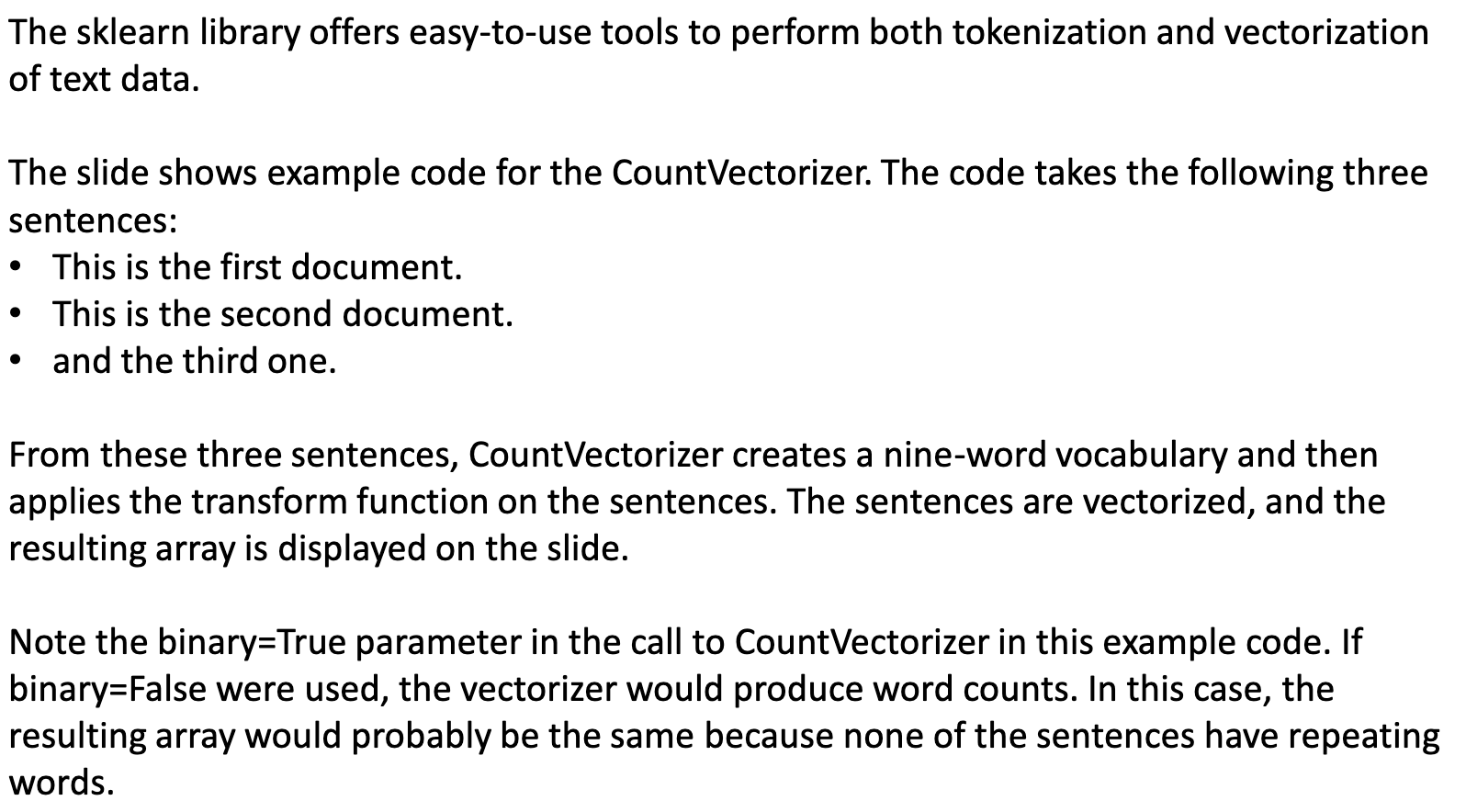
This method represents the text as a bag of words, and disregards the grammar and order of words. The steps are as follows:

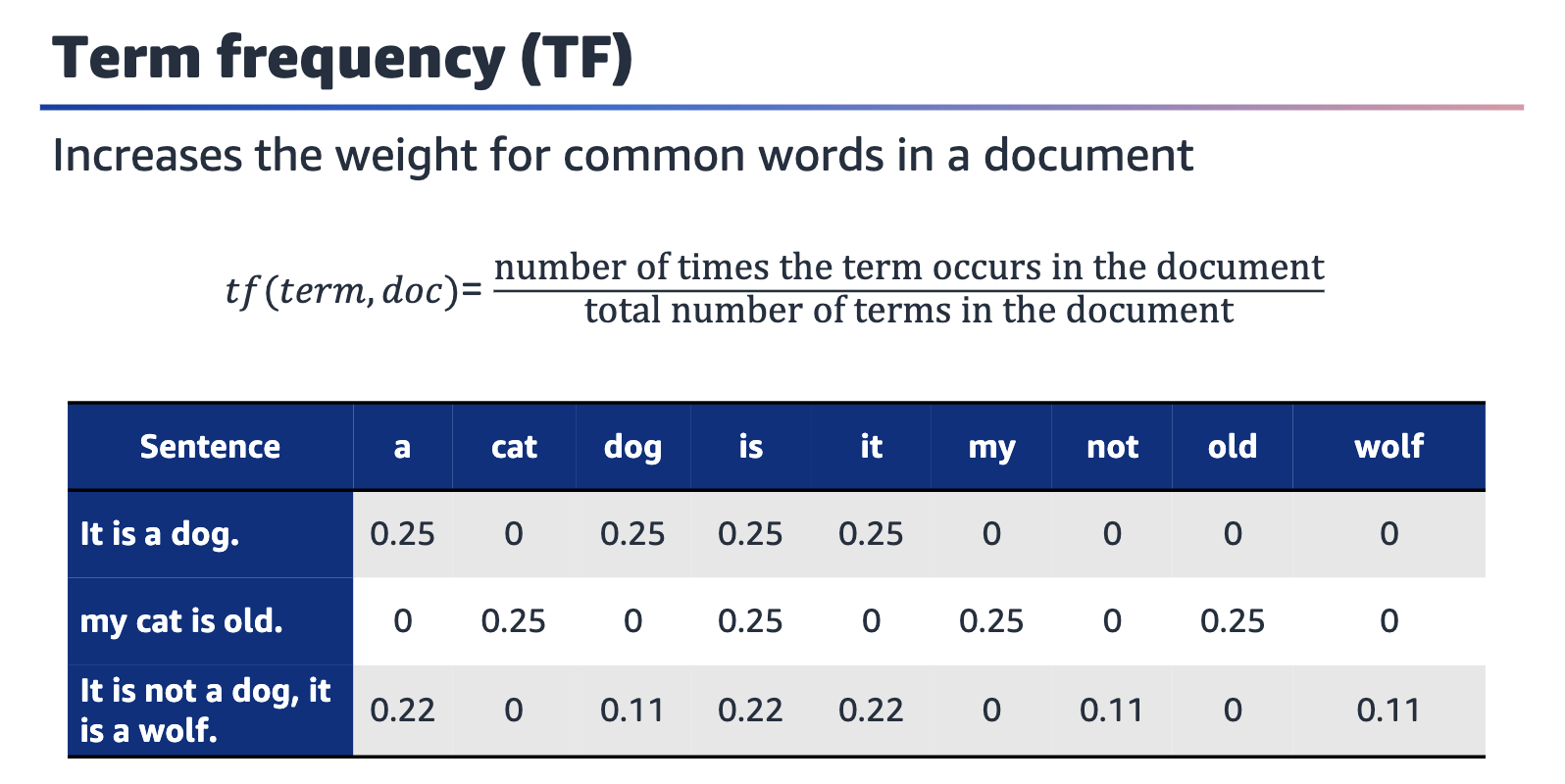
1. ﻿﻿﻿Create or borrow a vocabulary of words. If you don't borrow a vocabulary, you can create one by using the training data.
2. ﻿﻿﻿Create simple numerical vectors by calculating the occurrences of words in your text data. The vectors can be binary (whether the word occurs), simple word counts, or frequencies, which are normalized according to the total number of words in the text.



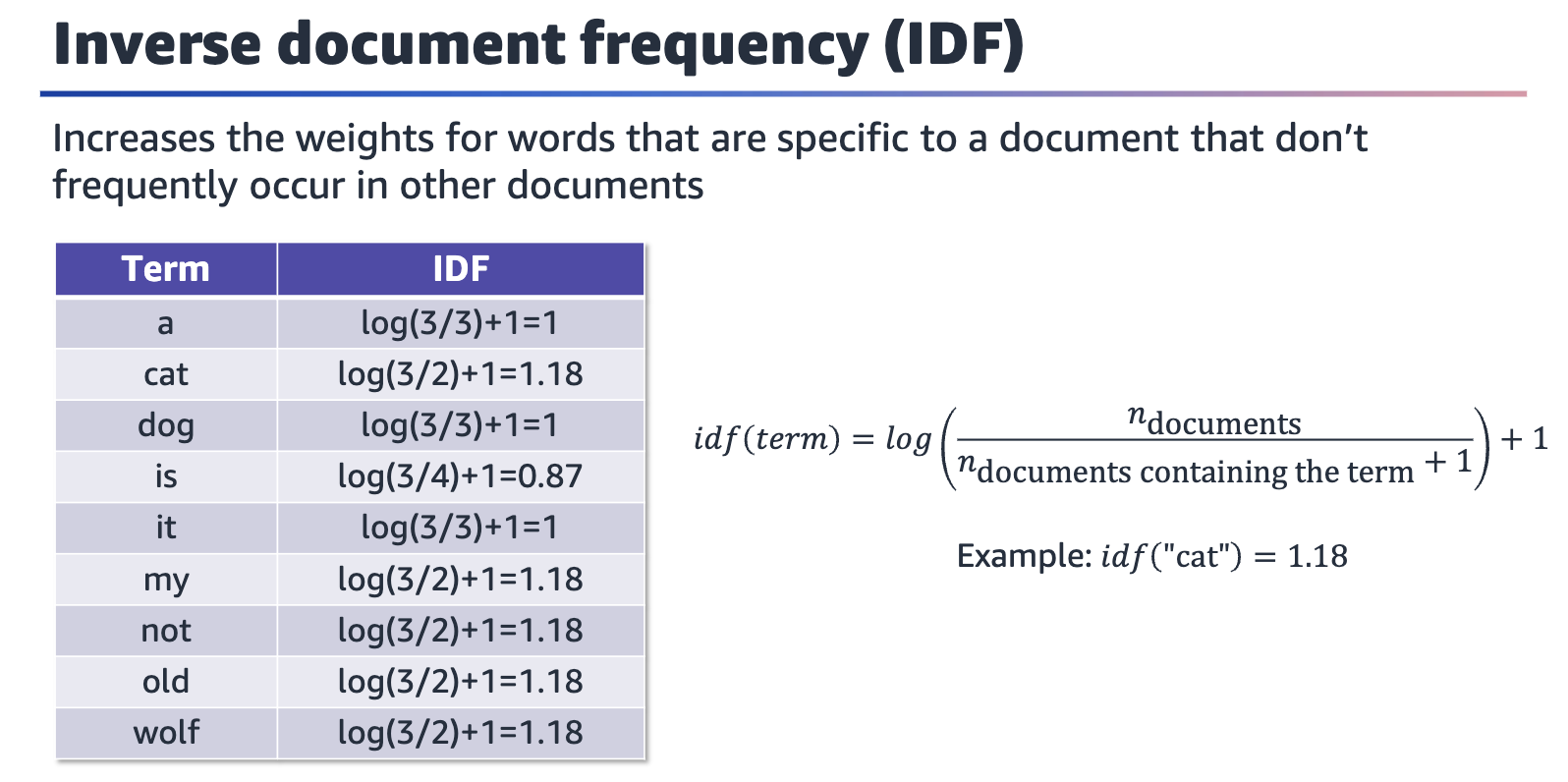










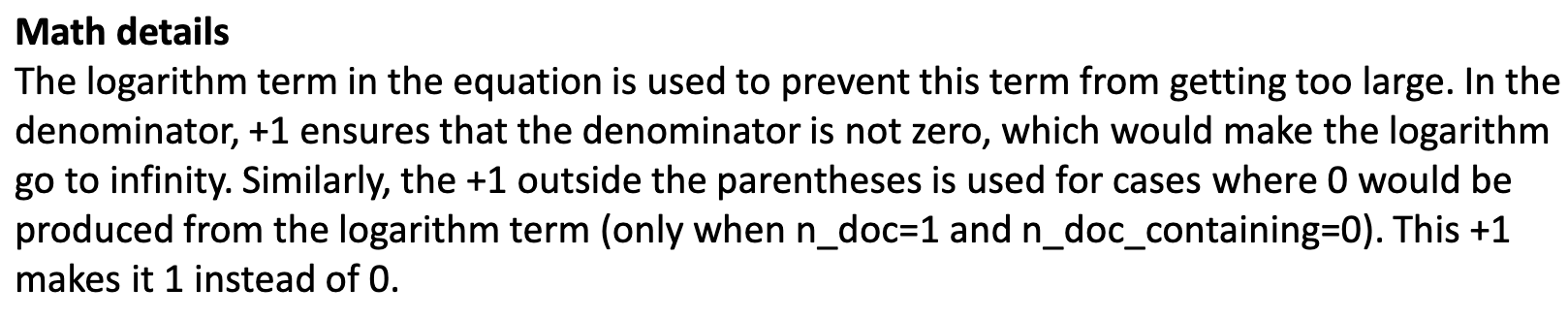


Now that you have seen how to convert text data into numerical data, one thing not yet discussed is the vocabulary words themselves. How frequently do you see the word "dog" or "plane" or "kid" across all documents? Why does this even matter? Sometimes, in text data rare words can be decisive about the overall meaning of a text.

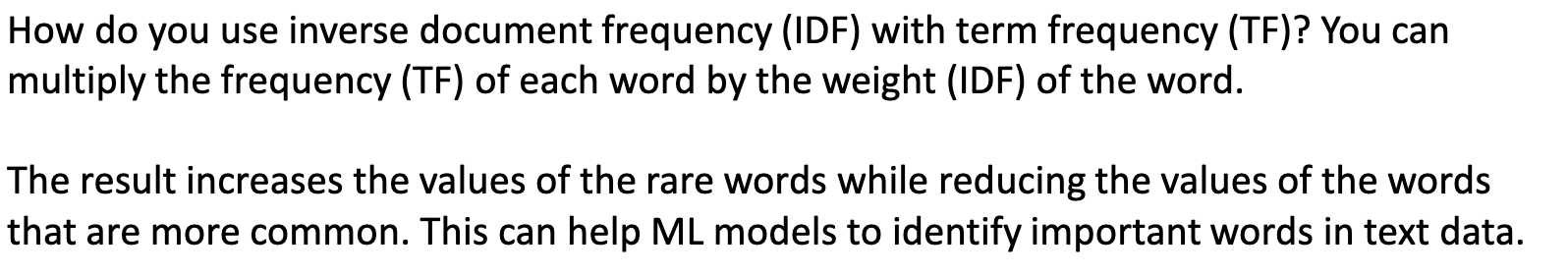
For example, imagine that you want to differentiate between documents about sports. You have documents about baseball, basketball, and soccer. Some words are common across all the documents; for example, supporter, field, player, team, ball, sport, jersey. Those words won't be helpful to identify a specific sport.

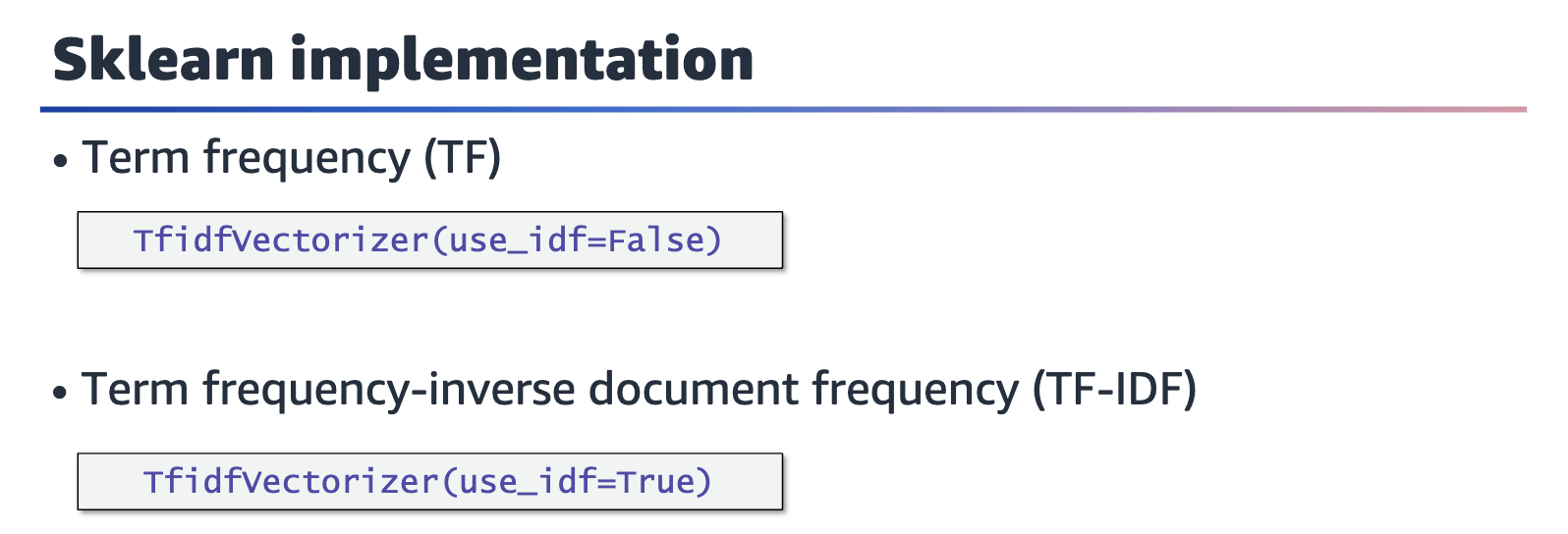
Some words aren't common and can be useful to differentiate between the sports. For example, goalie for soccer, pitcher for baseball, and point guard for basketball. Another example is galaxy for soccer, dodgers for baseball, and lakers for basketball. With these words, you can almost automatically know which sport a document refers to.

Inverse document frequency (IDF) assigns a weight to each word, depending on how frequent that word is across all documents. The weight is large if the word is rare and small if the word is common. This method incorporates the idea of word importance into the bag-of-words method.









You can use the sklearn library to perform TF vectorization as well as IDF vectorization with the same module.

For more information, see TfidfVectorizer in the sklearn documentation at [https://scikit-learn.org/stable/modules/generated/sklearn.feature extraction.text.TfidfVectorizer.html](https://scikit-learn.org/stable/modules/generated/sklearn.feature%20extraction.text.TfidfVectorizer.html).

