Embeddings

Estimated module length: 45 minutes

Learning objectives

- Visualize vector representations of word embeddings, such as word2vec.
- Distinguish encoding from embedding.
- Describe contextual embedding.

Prerequisites:

This module assumes you are familiar with the concepts covered in the following modules:

- Introduction to Machine Learning
- Linear regression
- Working with categorical data
- Neural networks

Imagine you're developing a food-recommendation application, where users input their favorite meals, and the app suggests similar meals that they might like. You want to develop a machine learning (ML) model that can predict food similarity, so your app can make high quality recommendations ("Since you like pancakes, we recommend crepes").

To train your model, you curate a dataset of 5,000 popular meal items, including borscht, hot dog, salad, pizza, and shawarma.

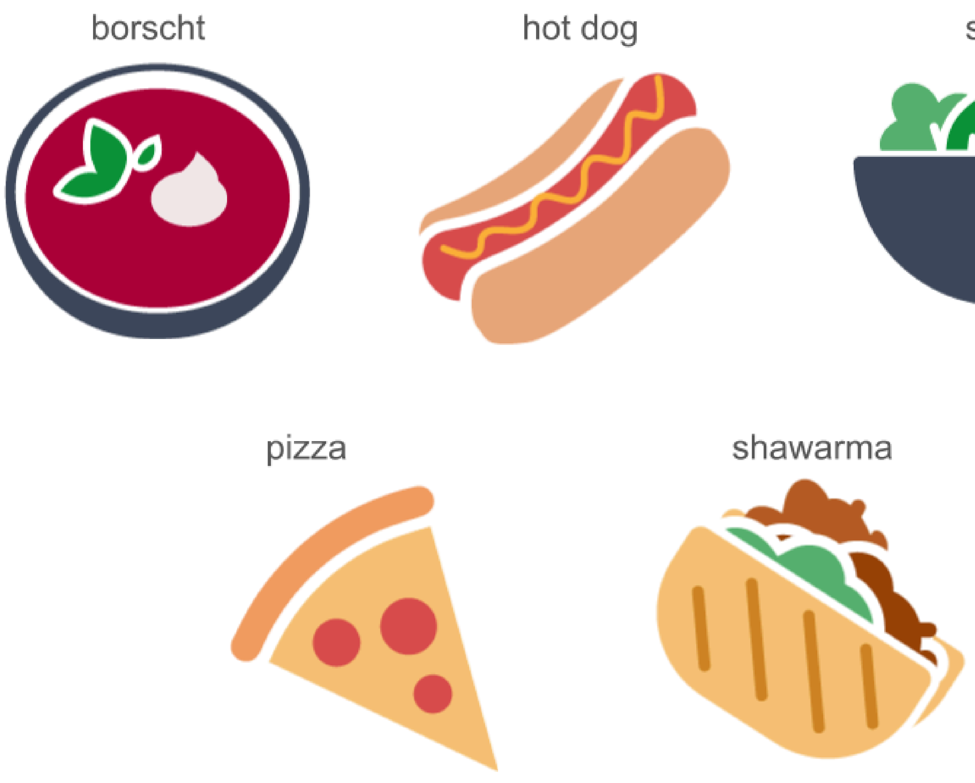

Figure 1. *Sampling of meal items included in the food dataset.*

You create a `meal` feature that contains a **one-hot encoded** representation of each of the meal items in the dataset. **Encoding** refers to the process of choosing an initial numerical representation of data to train the model on.

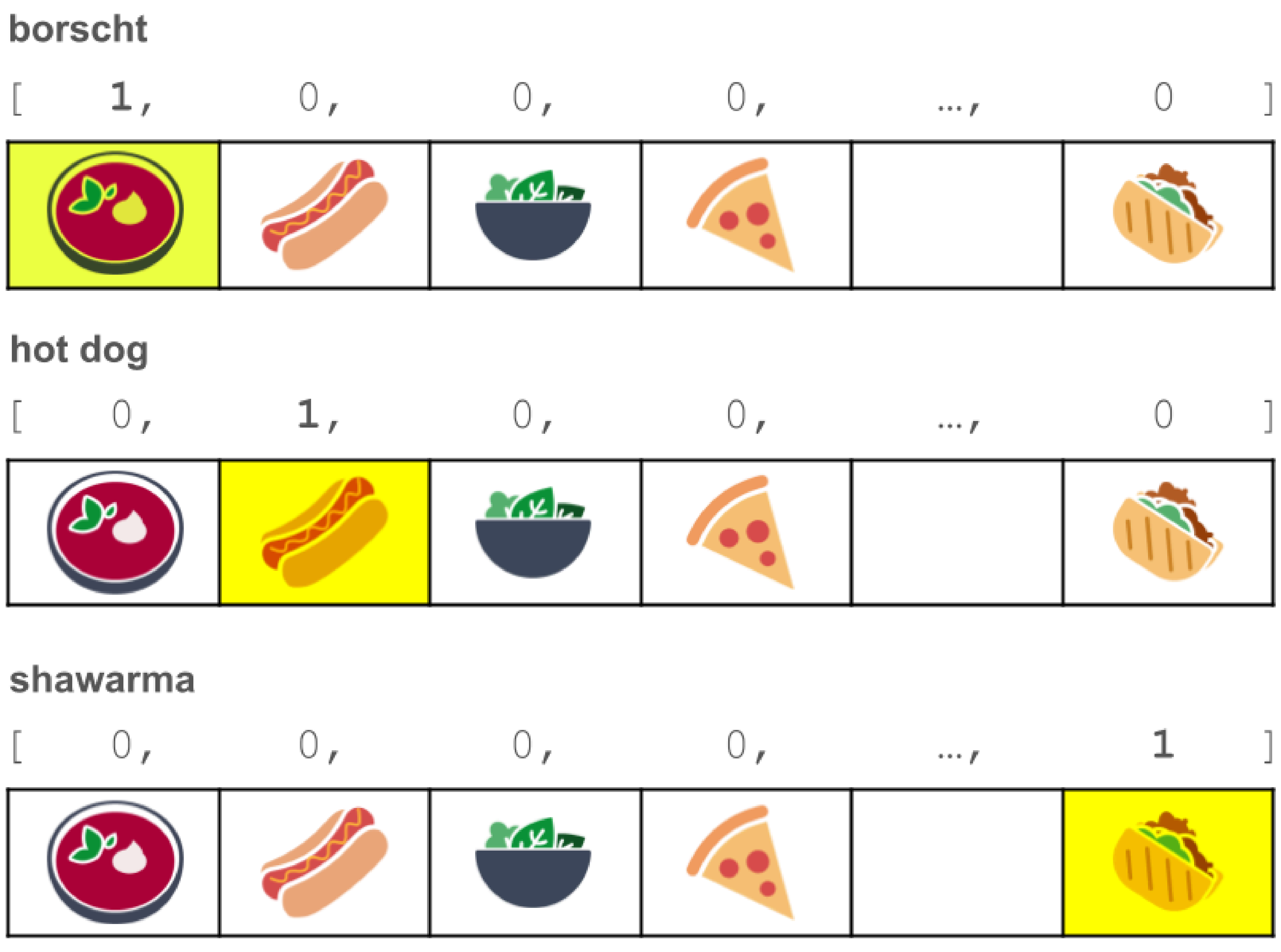

Figure 2. *One-hot encodings of borscht, hot dog, and shawarma. Each one-hot encoding vector has a length of 5,000 (one entry for each menu item in the dataset). The ellipsis in the diagram represents the 4,995 entries not shown.*

Pitfalls of sparse data representations

Reviewing these one-hot encodings, you notice several problems with this representation of the data.

- **Number of weights.** Large input vectors mean a huge number of **weights** for a **neural network**. With M entries in your one-hot encoding, and N nodes in the first layer of the network after the input, the model has to train MxN weights for that layer.
- **Number of datapoints.** The more weights in your model, the more data you need to train effectively.
- **Amount of computation.** The more weights, the more computation required to train and use the model. It's easy to exceed the capabilities of your hardware.
- **Amount of memory.** The more weights in your model, the more memory that is needed on the accelerators that train and serve it. Scaling this up efficiently is very difficult.
- **Difficulty of supporting on-device machine learning (ODML).** If you're hoping to run your ML model on local devices (as opposed to serving them), you'll need to be focused on making your model smaller, and will want to decrease the number of weights.

In this module, you'll learn how to create **embeddings**, lower-dimensional representations of sparse data, that address these issues.

Key terms:

- One-hot encoding
- Neural network
- Weight

Last updated 2025-07-02 UTC.