- Chapter 2 is designed to introduce the fundamentals of language modeling. It covers how to convert words and documents into numerical formats, explains the basics of language modeling, and presents count-based models as an initial architecture. Additionally, it details techniques for measuring language model performance.
- **2.1 Bag of Words (BoW)**
- The chapter begins by introducing the **Bag of Words (BoW)** method, a common and effective approach for converting a collection of documents (corpus) into feature vectors.
- **Process:** BoW involves two key steps:

1. **Creating a vocabulary:** Listing all unique words in the corpus, typically after removing punctuation, converting to lowercase, and eliminating duplicates.
2. **Vectorizing documents:** Converting each document into a feature vector where each dimension represents a word from the vocabulary, indicating its presence, absence, or frequency. The result can be organized into a **document-term matrix (DTM)**, where rows are documents and columns are tokens.

- **Tokenization and Subwords:** The process of splitting documents into indivisible parts is called **tokenization**, and each part is a **token**. While words are a common token type, **subwords** can be used to manage vocabulary size, especially for languages with many word forms.
- **Application in Classification:** The chapter demonstrates how these feature vectors can be used to train a neural network to predict a document's topic, framing it as a **multiclass classification** problem. It introduces **softmax activation function** for handling three or more classes, which maps raw neural network outputs (**logits**) to "probability scores" that sum to 1. The corresponding **cross-entropy loss** function is used to measure how well predicted probabilities match the true distribution (often a **one-hot vector**).
- **Limitations:** BoW has notable limitations: it fails to capture **token order or context**, meaning sentences with different meanings but the same words yield identical representations (e.g., "the cat chased the dog" vs. "the dog chased the cat"). It also struggles with **out-of-vocabulary (OOV)** words and treats synonyms as distinct terms.
- **2.2 Word Embeddings**
- To overcome BoW limitations, **word embeddings** are introduced as a superior approach. Instead of sparse one-hot vectors, word embeddings represent words as **dense vectors** (lower-dimensional representations with mostly non-zero values).

- **Semantic Similarity:** A key advantage is that **semantically similar words have embeddings that exhibit high cosine similarity**. This allows models to recognize and process words with similar meanings more efficiently.
- **Learning Embeddings:** These embeddings are learned from vast unlabeled datasets. The chapter explains **Word2vec**, specifically the **skip-gram formulation**, where a model predicts missing words from their surrounding context. This training helps the model learn semantic relationships. The **cross-entropy loss** is also used here, adapting to a large number of "classes" (vocabulary words).
- **Dimensionality Reduction:** Word embeddings offer **dimensionality reduction**, compressing word representations into smaller vectors (e.g., 100-1000 dimensions). The chapter illustrates how techniques like **Principal Component Analysis (PCA)** can project high-dimensional embeddings into 2D, revealing semantic connections.
- **Impact:** The ability of word2vec embeddings to support meaningful arithmetic operations (e.g., "king – man + woman ≈ queen") was a pivotal moment, showing that neural networks could encode semantic relationships, paving the way for advanced language models.
- **2.3 Byte-Pair Encoding (BPE)**
- BPE is presented as a **tokenization algorithm** that addresses OOV words by breaking them into smaller units called **subwords**.
- **Algorithm:** BPE iteratively merges the most frequent adjacent symbol pairs (characters or subwords) into new subword units until a target vocabulary size is reached.
- **Practicality:** While initially a data compression technique, its adaptation for NLP helps keep vocabulary sizes manageable, especially for morphologically rich languages. The chapter provides a Python implementation of the BPE algorithm's core components.
- **Efficiency:** Modern LLMs often use subwords and have large vocabularies (e.g., over 100,000 tokens), making efficient tokenization crucial. Optimized versions of BPE use caching and precomputed data structures for faster processing.
- **2.4 Language Model**
- A **language model** is formally defined as a model that predicts the next token in a sequence by estimating its conditional probability based on previous tokens. It assigns a probability to all possible next tokens in the vocabulary, which sum to 1, forming a valid discrete probability distribution.
- **Types:**
- **Autoregressive (Causal) Language Models:** Predict an element in a sequence using only its predecessors. These excel at text generation and include Transformer-based chat language models.

- **Masked Language Models:** Predict intentionally masked tokens within sequences, utilizing both preceding and following context (e.g., BERT). The book primarily focuses on autoregressive models.
- **2.5 Count-Based Language Model**
- The chapter implements a simple **count-based language model**, specifically a **trigram model** (n=3).
- **Probability Calculation:** The probability of a token is calculated based on the two preceding tokens using the **maximum likelihood estimate (MLE)**, which is the relative frequency of an n-gram.
- **Addressing Zero Probabilities:** To handle unseen n-grams that would otherwise have zero probability, the model uses **backoff**, defaulting to lower-order n-grams if a higher-order one is not observed. **Add-one smoothing (Laplace smoothing)** is introduced to ensure all tokens, including unseen ones, receive a small, non-zero probability.
- **Training and Evaluation:** The chapter details the `CountLanguageModel` class and its `train` and `predict_next_token` methods, using a portion of the **Brown Corpus** as training data. It emphasizes the importance of **partitioning data into training and test sets** to control **overfitting** and evaluate generalization.
- **Limitations:** Count-based models have several drawbacks: they generally use **word-tokenized corpora** and small n-gram sizes (up to n=5) due to memory constraints. They **cannot handle out-of-vocabulary (OOV) words** effectively and are **unable to capture long-range dependencies** in language. Moreover, their n-gram counts are fixed, making **adaptation for downstream tasks difficult without retraining**. These limitations highlight why neural network-based language models have largely replaced them.
- **2.6 Evaluating Language Models**
- This section covers crucial metrics and techniques for assessing language model performance:
- **Perplexity:** A widely used metric that measures how well a model predicts text. **Lower perplexity values indicate a better model**. It is defined as the exponential of the average **negative log-likelihood (NLL)** per token in the test set. Perplexity can also be understood as the geometric mean of inverse probabilities, indicating the "weighted average factor by which the model is 'perplexed'". A perplexity of 10 means the model is as uncertain as if it had to choose uniformly between 10 possibilities. The chapter provides a detailed example of perplexity calculation.
- **ROUGE (Recall-Oriented Understudy for Gisting Evaluation):** Used for finetuned models, ROUGE evaluates text quality by measuring overlaps (tokens or n-grams) between generated and reference texts.
- **ROUGE-N:** Evaluates overlap of n-grams, with N indicating length.

- **ROUGE-1:** Measures unigram (single token) overlap, focusing on **recall**.
- **ROUGE-L:** Relies on the **longest common subsequence (LCS)**, which captures word order and sentence structure.
- **Limitations:** ROUGE measures lexical overlap but **not semantic similarity or factual correctness**.
- **Human Evaluation:** Crucial for assessing qualities like fluency and accuracy that automated metrics miss.
- **Likert Scale Ratings:** Raters judge quality on a symmetric scale (e.g., -2 to 2). Challenges include central tendency bias and rater inconsistency.
- **Pairwise Comparison:** Two outputs are evaluated side-by-side, and the better one is chosen. This method is often analyzed statistically using the **Elo rating system**. Elo ratings quantify relative model performance based on "wins" and "losses" in direct comparisons.