edureka!
a Veranda Enterprise | ILLINOIS TECH

POST GRADUATE PROGRAM IN GENERATIVE AI AND ML

Natural Language Processing

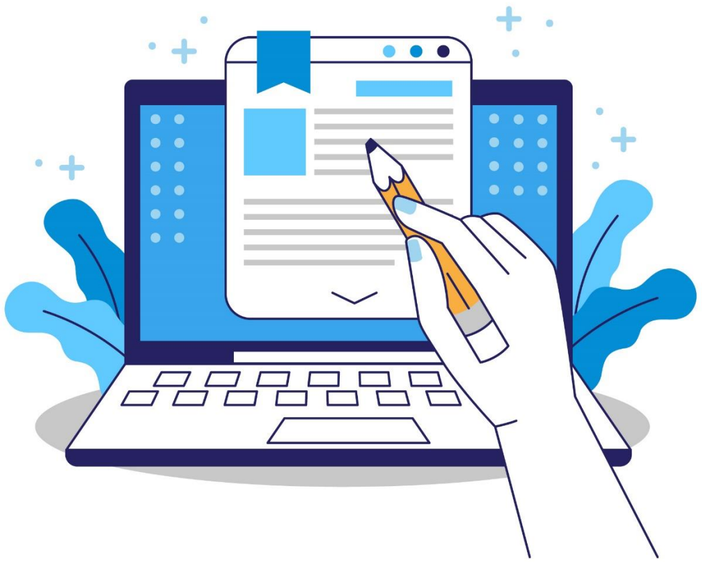
edureka!
a Veranda Enterprise

ILLINOIS TECH

Tokenization and Text Encoding

Topics

- Fundamentals of Subword
- Tokenization Techniques
- Tokenization Algorithms: WordPiece, SentencePiece, and N-Grams
- Dynamic Tokenization Strategies
- Tokenization in Transformer-Based Models
- Tokenization for Low-Resource and Morphologically Rich Languages
- Character-Level and Byte-Level Embeddings
- Hybrid Embedding Models and Character-Level CNNs in Transformers
- Sentence Embeddings and Semantic Similarity Techniques
- Applications of Sentence Embeddings in NLP Tasks



Learning Objectives

By the end of this lesson, you will be able to:

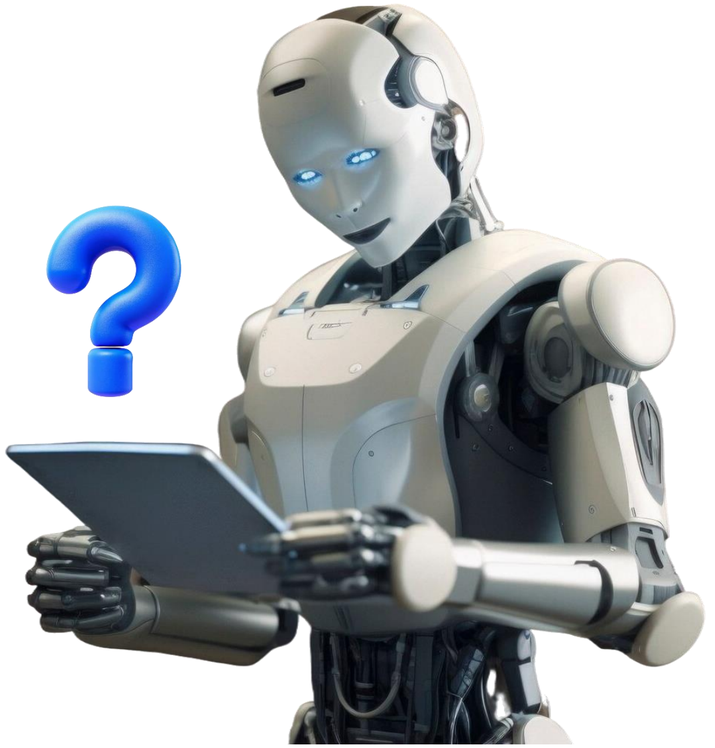
- Apply subword tokenization using BPE, Unigram LM, and byte-level methods.
- Implement WordPiece, SentencePiece, and N-gram-based tokenization techniques.
- Use dynamic tokenization in adaptive NLP settings.
- Use tokenizers in transformer models like BERT and GPT for downstream tasks.
- Generate character and hybrid embeddings using CNNs, RNNs, and Transformers.
- Create and apply sentence embeddings for semantic search, chatbots, and clustering.

Introduction to Subword Tokenization

Why Machines Don’t Read Like You Do?

“Ever wondered how ChatGPT understands the word ‘unbelievably’?”

unbelievably

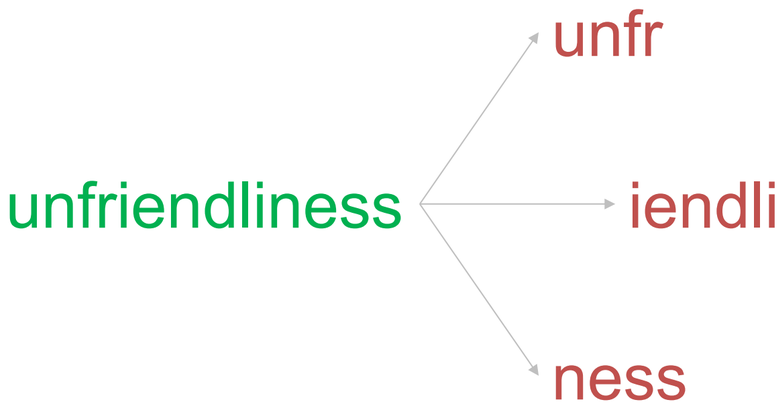


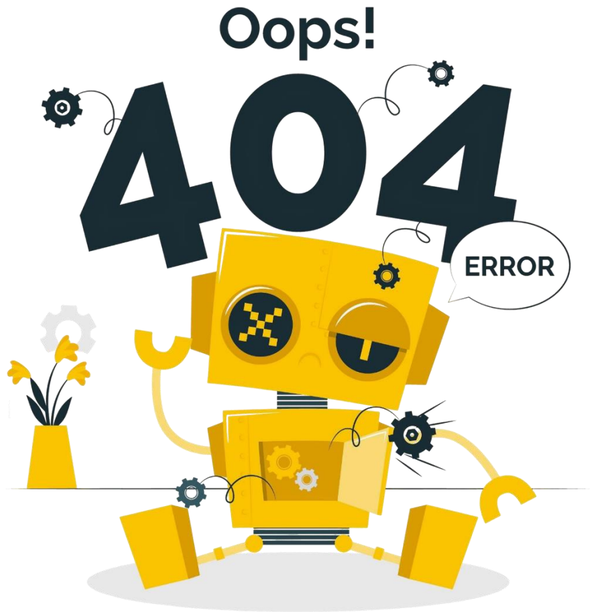
Problem – The Real Challenge in NLP

"In English, there are over 170,000 words. But AI doesn't memorize them all."

Naïve Tokenization Fails

"Basic Tokenizers split by spaces."



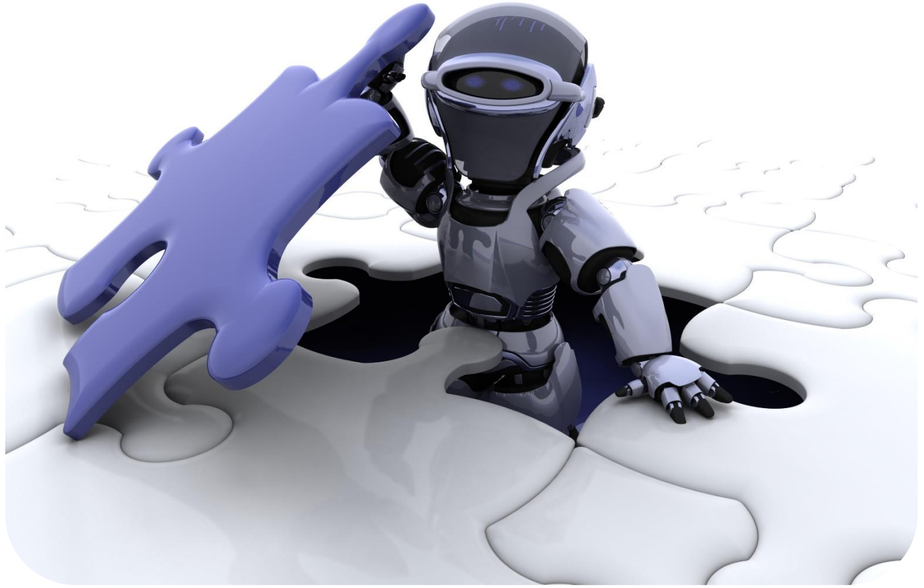


Solution – Subword Tokenization

"What if AI learned common pieces of words?"

un + friend + li + ness

unfriendliness



WordPiece and SentencePiece Algorithms

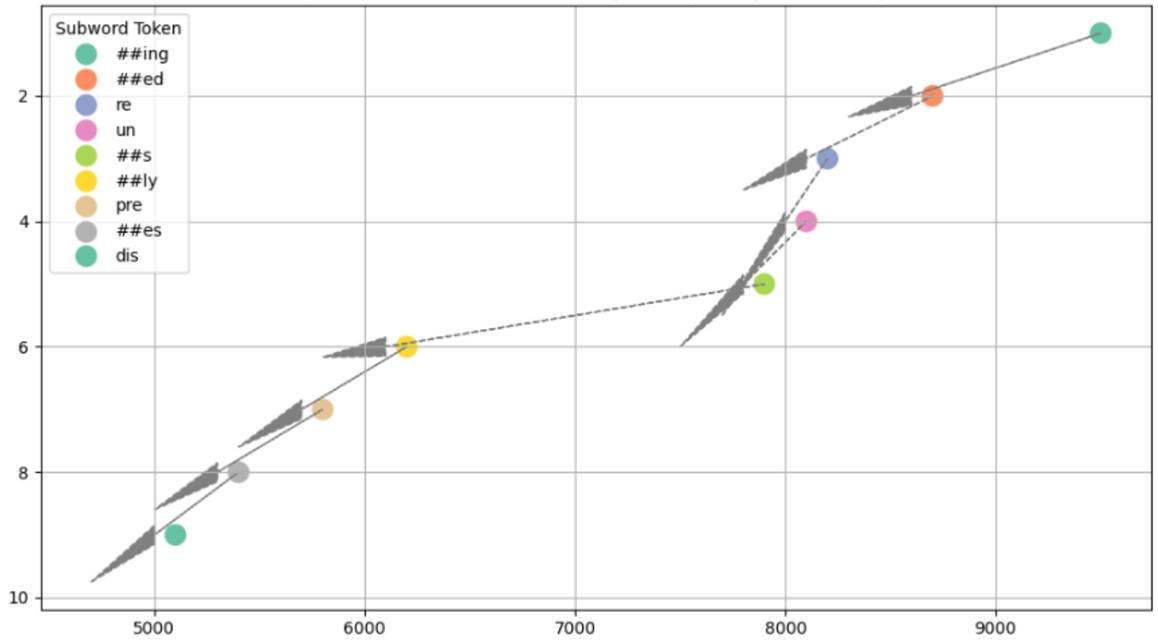
WordPiece – DNA of Language

"Basic Tokenizers split by spaces."

Each base pair is a subword: ##ing, ##ed, ##ly, un, play, re, etc.

un + believ + able → **unbelievable**

play + ##ing → **playing**

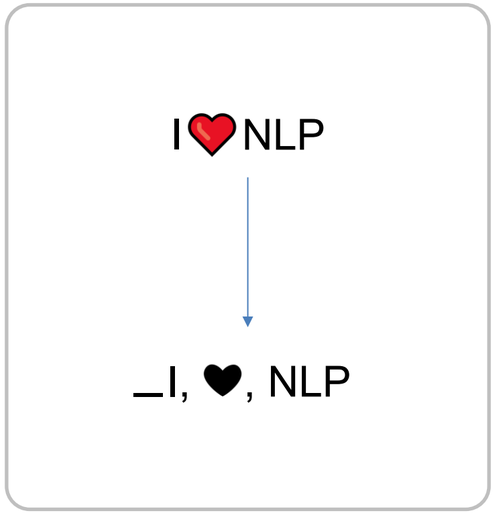


SentencePiece – The Radical Rebel

"Basic Tokenizers split by spaces."

私はNLPが大好きです (I love NLP)

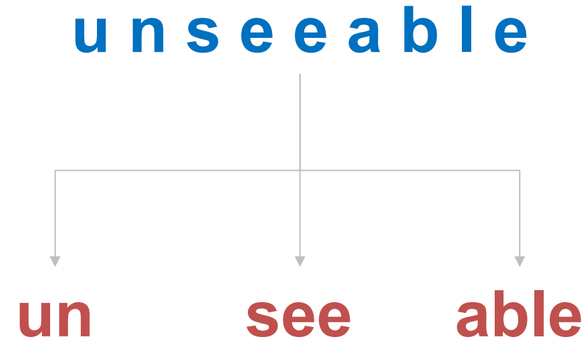
_私は　NLP　が　大好き　です



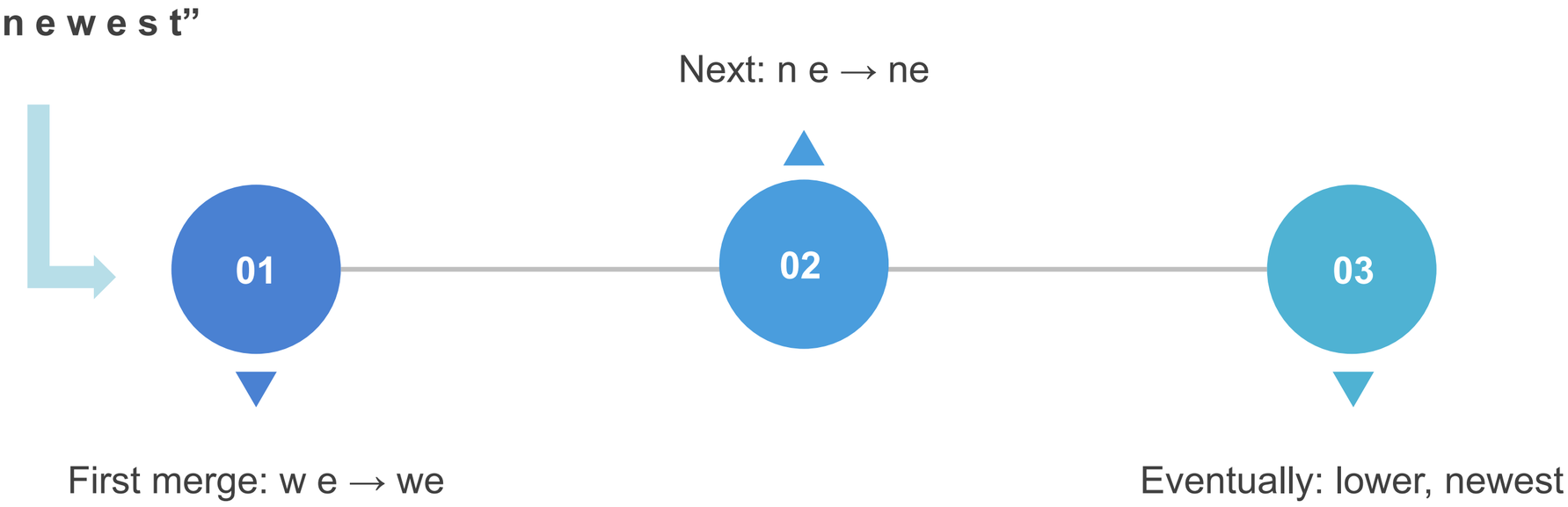
Byte-Pair Encoding (BPE) and Unigram Language Models

BPE — “Build It from Scratch”

Instead of memorizing whole words, it learns to build words. Byte-Pair Encoding is like Lego for language.



How BPE Learns?



N-Gram-Based Approaches in Tokenization

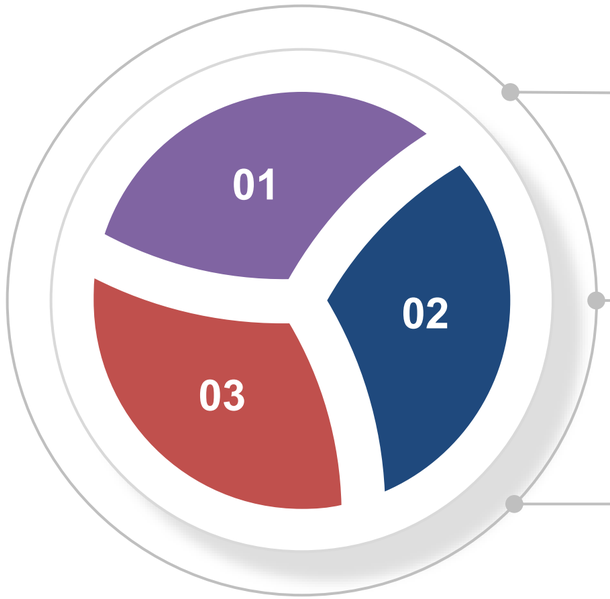
Unigram — The Probability Wizard

The Unigram model does things differently. It doesn't build from the bottom up — it chooses the best sequence.

friendlessness

- e! [friend][less][ness]
- e! [friendless][ness]
- e! [fri][endl][essness]

How Unigram Works



Start with a huge list of possible subwords.

Train a model to assign probabilities.

Keep only the best ones. Let the rest go.

N-Grams

Before AI learned to read subwords and context like a pro, it was just guessing based on the past using N-Grams.

“I love deep ____”

- **learning** (✅ likely)
- **water** (🌊 okay)
- **dish** (🍽️ what?!)

Tokenization in Transformer Models (BERT, GPT, RoBERTa)

Transformer Models: BERT, GPT, and RoBERTa

Transformer models rely on subword tokenization to handle vocabulary efficiently. Tokenization converts raw text into tokens (subwords or byte-pair units), which are mapped to IDs before model input.

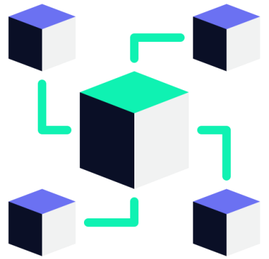
Model	Tokenizer Type	Special Features
BERT	WordPiece	Uses [CLS] and [SEP] tokens
GPT	Byte-Level BPE	Preserves whitespace and punctuation
RoBERTa	Byte-Level BPE	Uses <s> and </s> tokens, no segment IDs

Dynamic Tokenization Strategies

What is Dynamic Tokenization?

Input Text
(Adaptive Rules)

Tokenizer



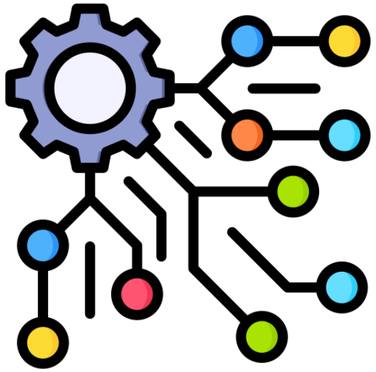
Tokens
(Vary by Context)

Dynamic Tokenization Strategies

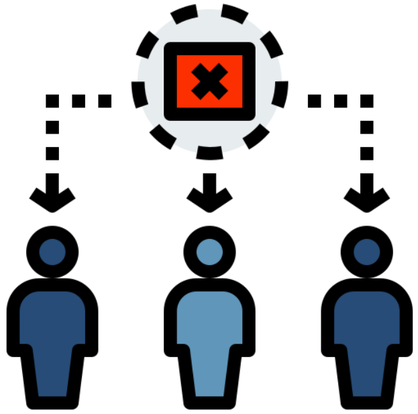
Strategy	Description
Subword Regularization	Randomly samples from multiple valid token splits during training for robustness.
BPE-Dropout	Drops some BPE merge rules at random to encourage diverse tokenization.
Dynamic Vocab Creation	Builds/updates vocabulary on-the-fly based on the input or domain.
Language-Aware Tokenization	Adjusts token splitting per language or script in multilingual settings.
Contextual Token Splitting	Uses model or embedding context to define token boundaries dynamically.

Tokenization for Low-Resource and Morphologically Rich Languages

Challenges in Tokenization

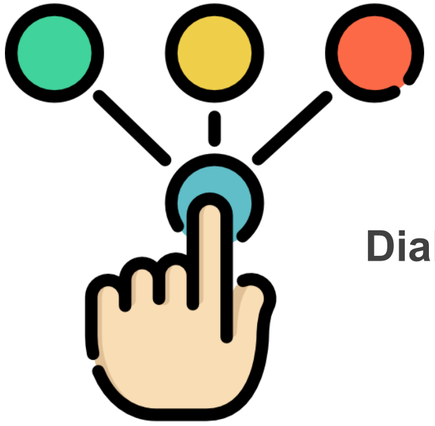


Complex Morphology



Lack of Resources

Word Segmentation Difficulty



Dialectal and Orthographic Variability

Techniques for Tokenization

Rule-Based Approaches

Multilingual Pretrained Models

Morphological Analyzers

Character-Level Tokenization

Subword Tokenization

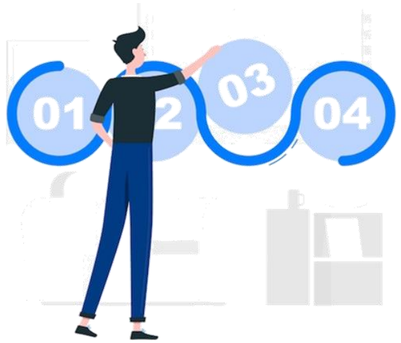
Neural Tokenization Models

Byte-Level Tokenization (Byte-Level BPE)

Byte-Level Tokenization

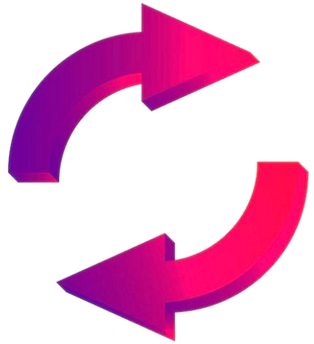
Byte Pair Tokenization (BPE) is a text tokenization technique that iteratively **merges the most frequent pairs** of characters or character sequences in to create subword units, enabling efficient handling of rare and unknown words.

Initialize Vocabulary

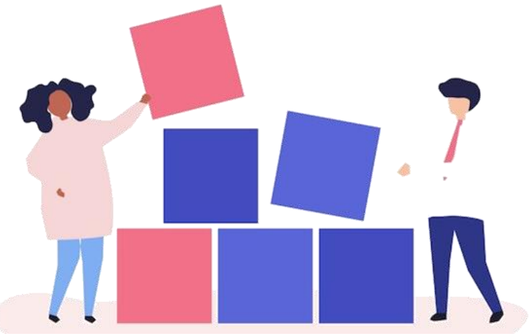


Count Pair Frequencies

Merge Most Frequent Pair



Repeat Merges

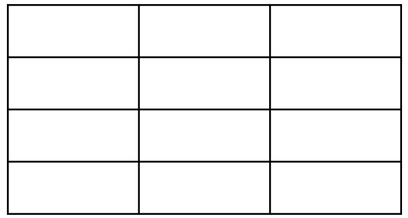


Tokenize Text Using Final Vocabulary

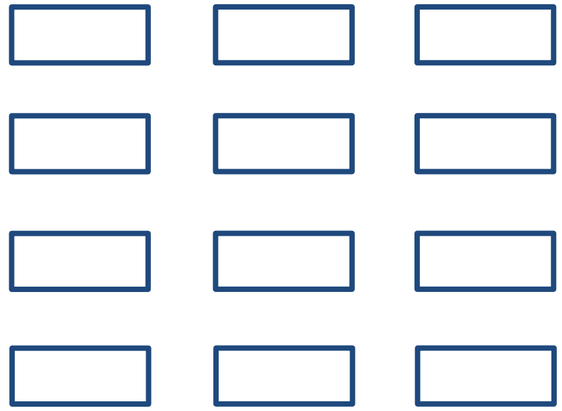
Adaptive Tokenization in Domain-Specific Contexts

Adaptive Tokenization

Adaptive Tokenization is a dynamic tokenization technique that **adjusts the way text is split into tokens** based on the input data's structure, context, or language, rather than relying on a fixed rule set.



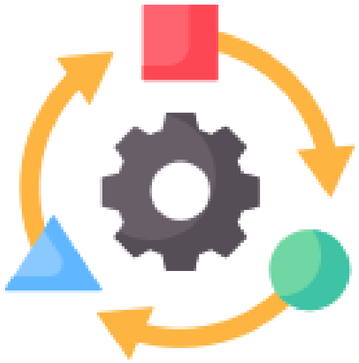
Data with patterns



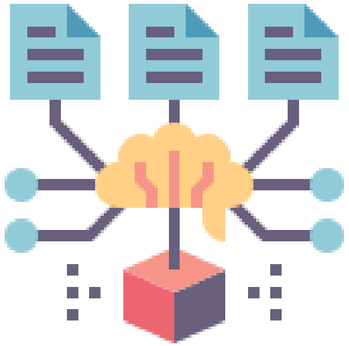
Tokenization on the basis of domain-specific terminology, linguistic patterns and application requirements

Types of Adaptive Tokenization

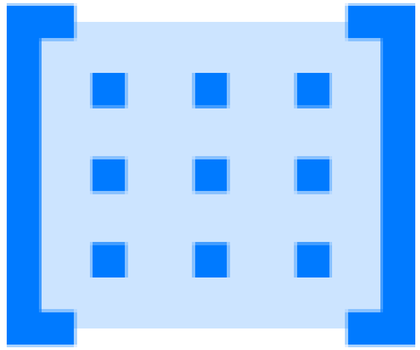
Rule-based Adaptive Tokenization



Subword Adaptive Tokenization



Model-based Adaptive Tokenization



Embedding-aware Tokenization

Core Adaptive Techniques

Domain-Optimized BPE Variants

Modern implementations enhance standard BPE through:

Priority-based initialization

AdaptBPE modifies BPE's merge rule creation by prioritizing longest string matches from domain vocabularies before character-level operations. This approach improves classification accuracy by 3.57% and summarization ROUGE-L by 1.87% in financial/medical domains.

Dynamic vocabulary expansion

The KL3M framework constructs specialized vocabularies through iterative co-occurrence analysis of domain corpora, achieving 128K token vocabularies with 40% higher coverage for legal terminology

Core Adaptive Techniques (contd.)

Hybrid Vocabulary Architecture

Task-adaptive tokenization employs a three-stage process:

- **Vocabulary merging:** Integrates domain tokens into existing vocabularies through positional encoding offsets
- **Domain lexicon extraction:** Identifies high-value n-grams using pointwise mutual information (PMI) scoring
- **Segmentation sampling:** Generates multiple valid tokenizations with probabilities weighted by domain relevance

It enables 60% reduction in sequence length for mental health Q&A tasks while improving answer quality

Character-Level Embeddings with CNNs and RNNs

Why Character-Level Embeddings?

Words are made of characters

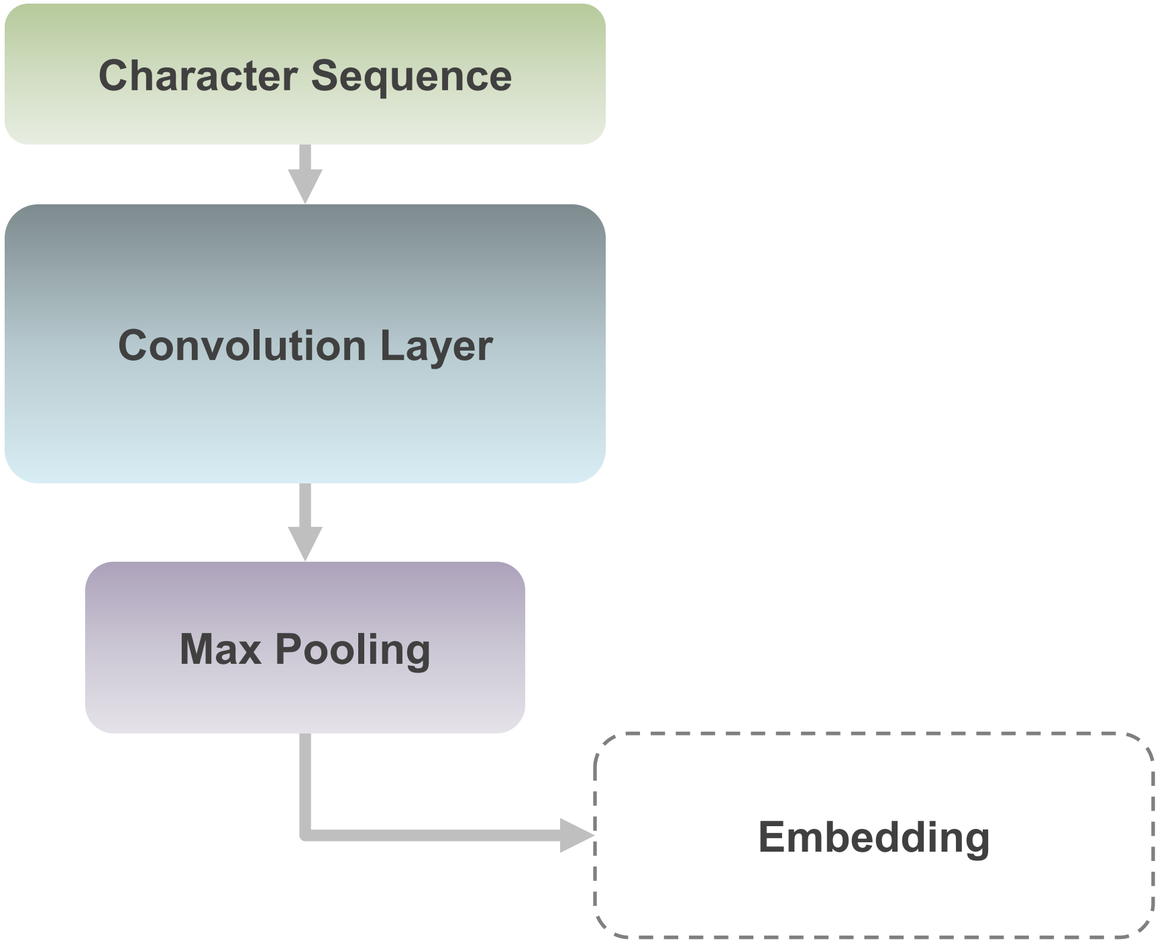
Handles typos, misspellings, rare words

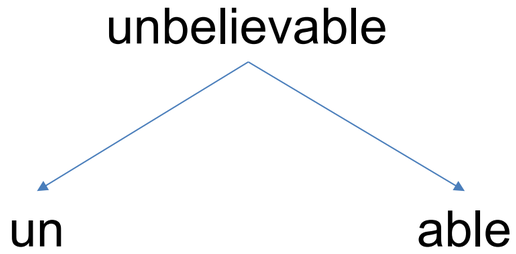
Useful for morphologically rich languages

playing → **[‘P’, ‘l’, ‘a’, ‘y’, ‘i’, ‘n’, ‘g’]**

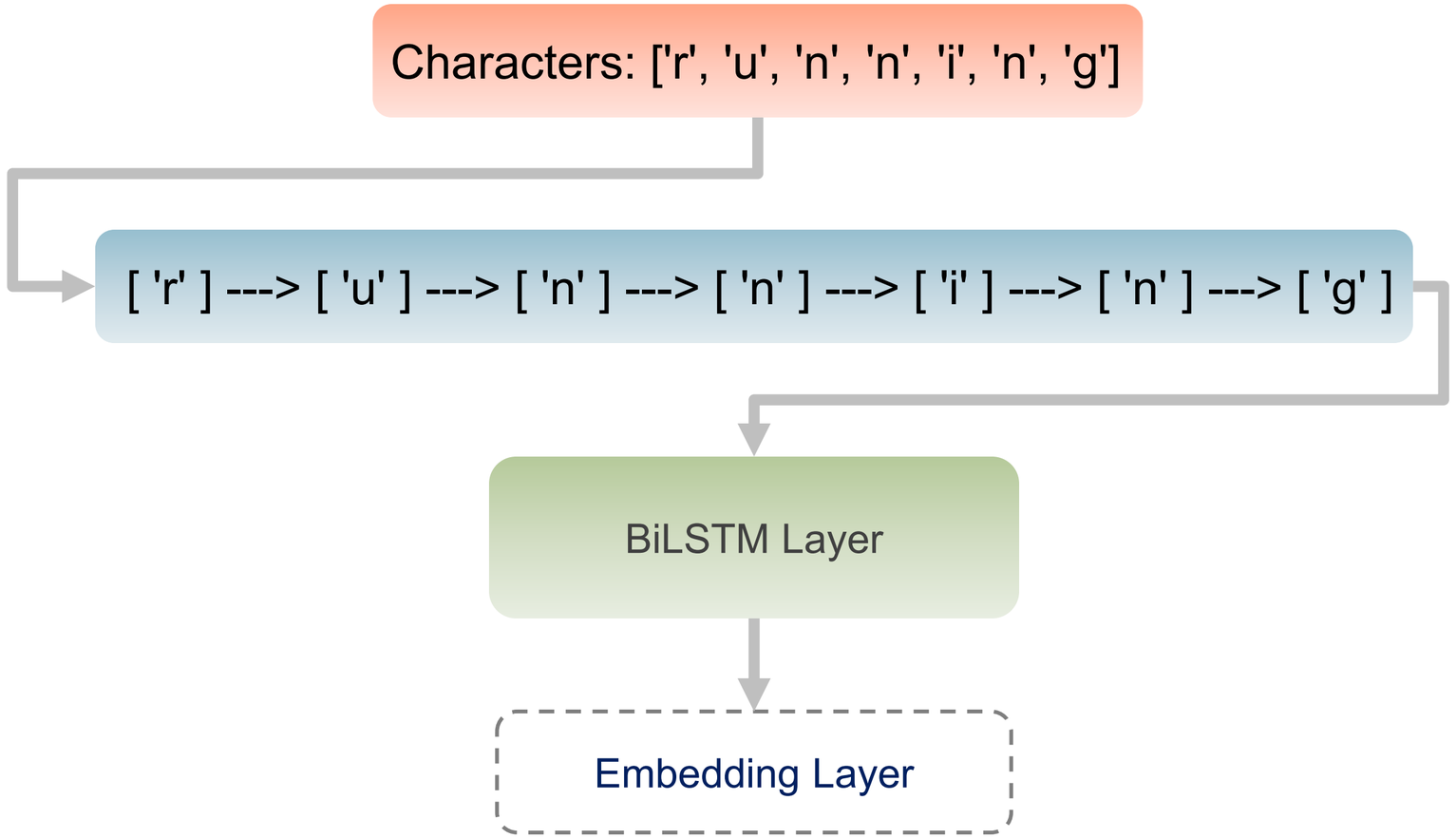
color	colour

How CNNs Process Characters





How RNNs Process Characters



FastText: Subword Embeddings and Their Utility

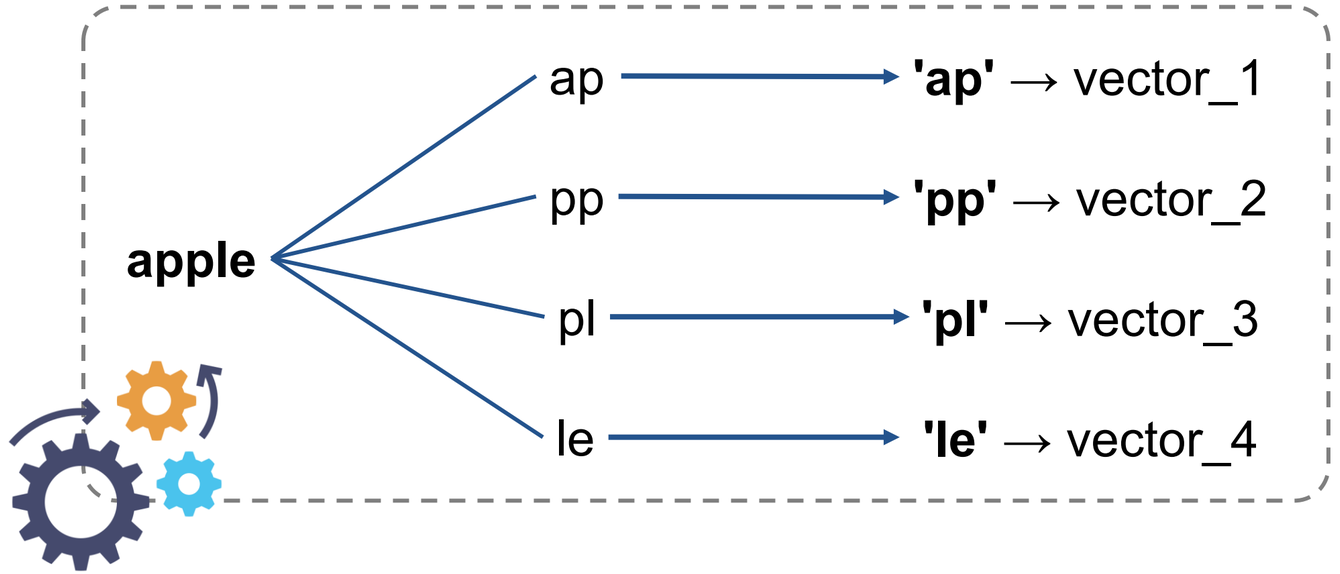
How FastText Works - Subword N-Grams

FastText splits words into smaller parts (subwords) like n-grams.

Example: running

n-grams: ['ru', 'un', 'nn', 'ni', 'in', 'ng']

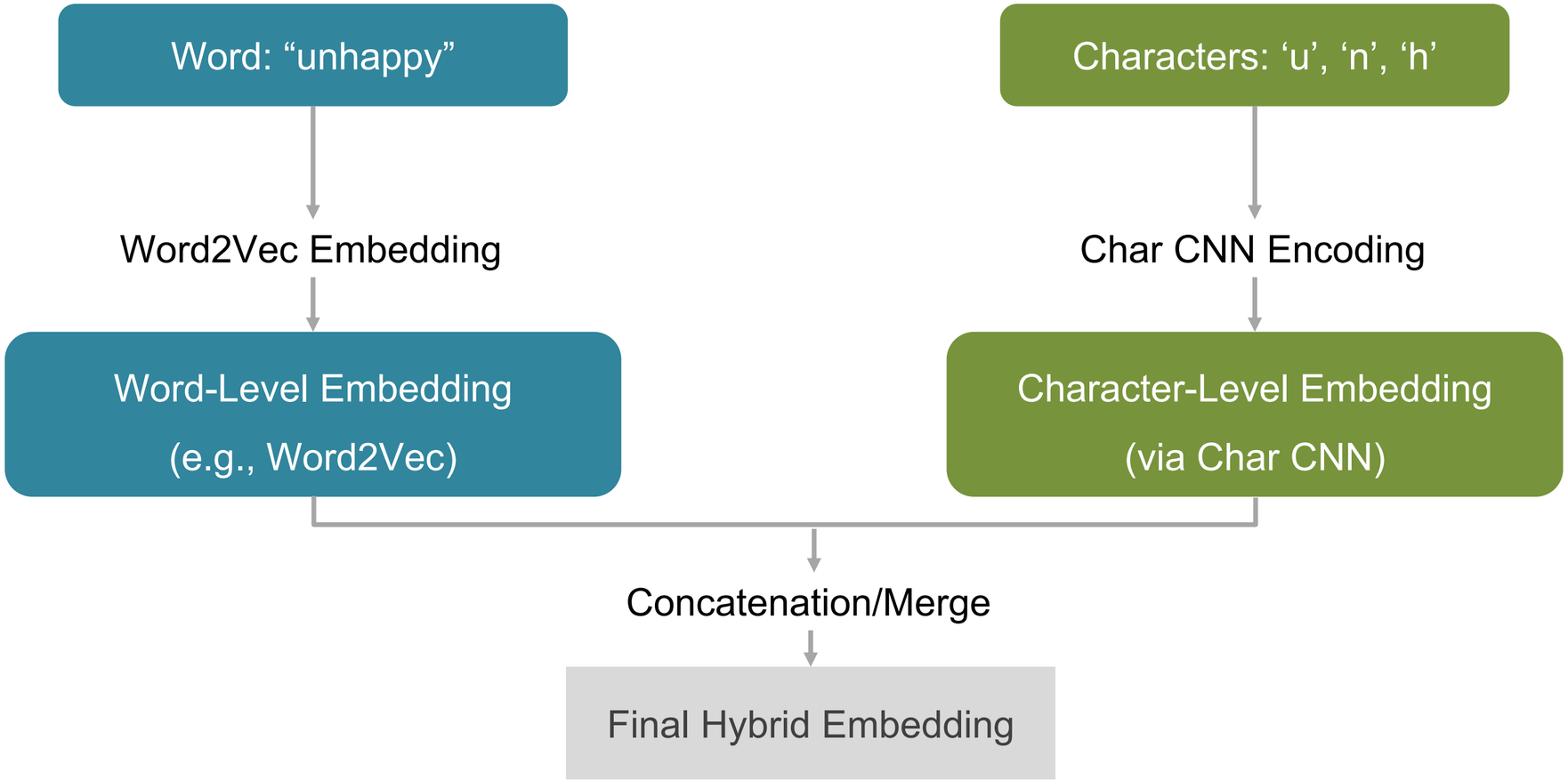
Fast Text's Training Process



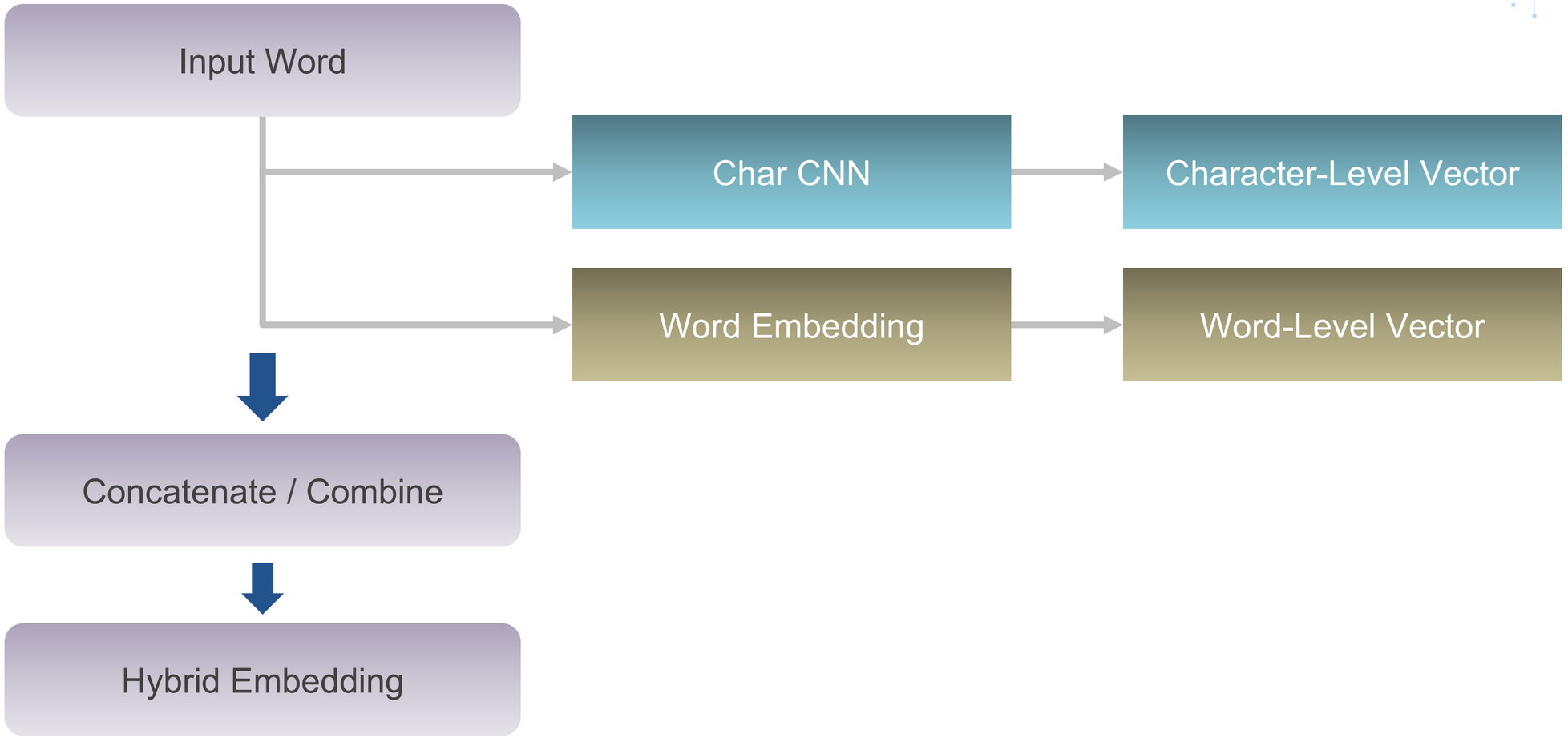
Final word embedding for "apple" = vector_1 + vector_2 + vector_3 + vector_4

Hybrid Embeddings: Combining Character and Word Representations

Why Combine Word + Character Embeddings?



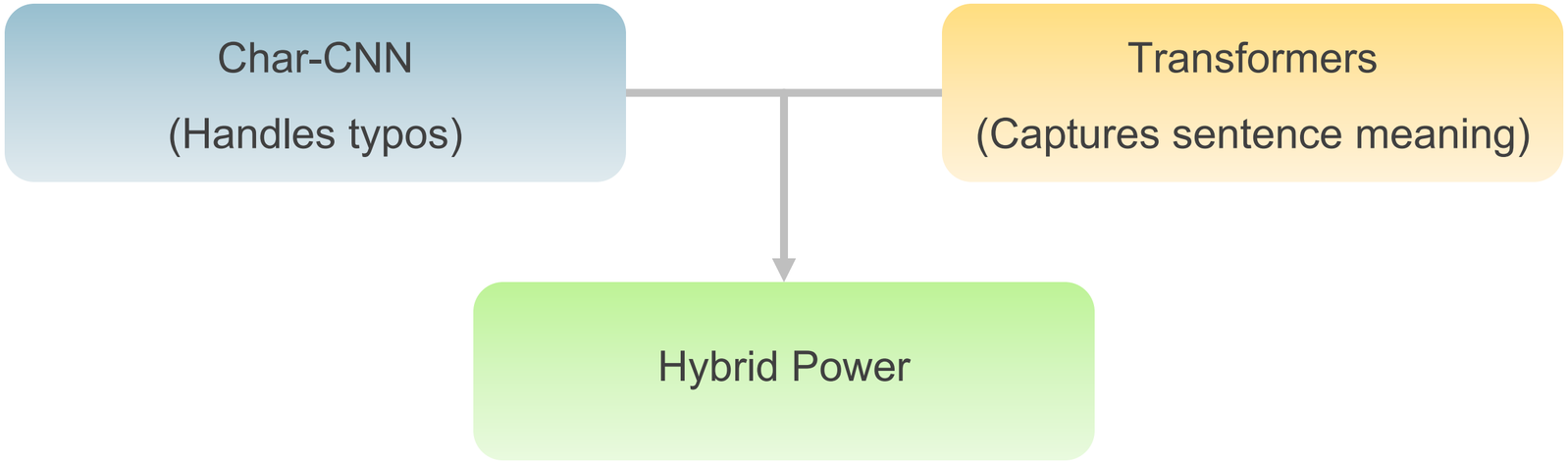
Hybrid Architecture



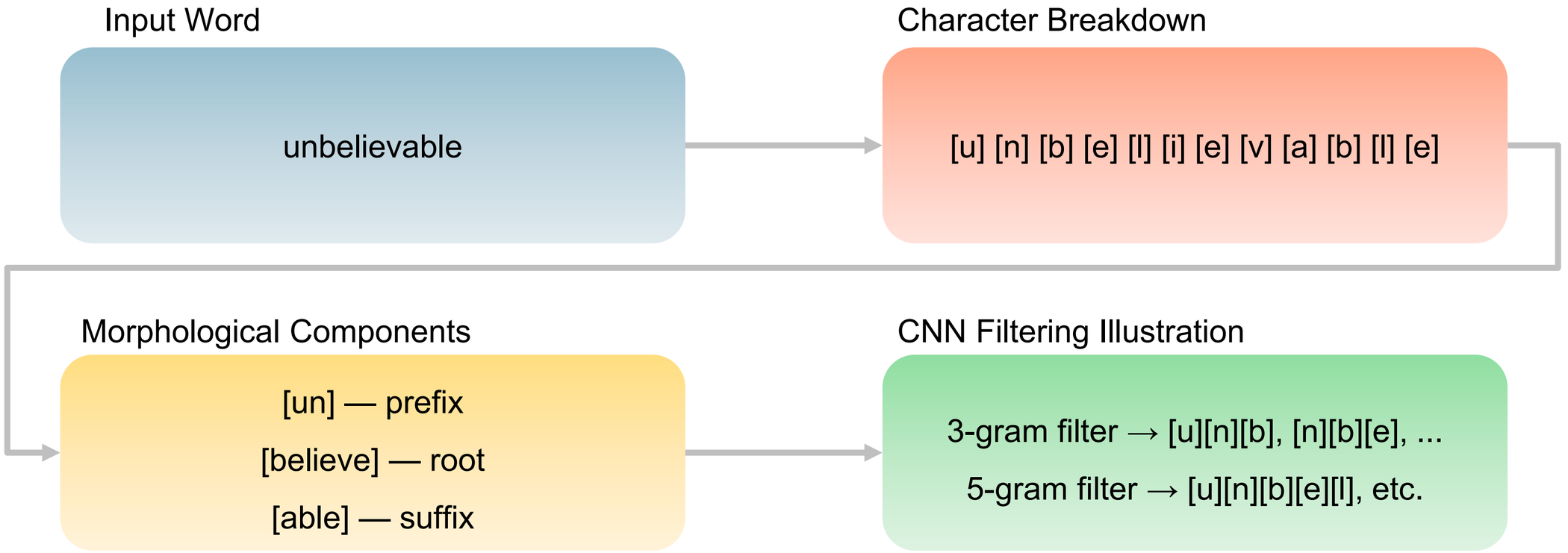
Hybrid Models: Character-CNNs Integrated with Transformers

What are Hybrid Models?

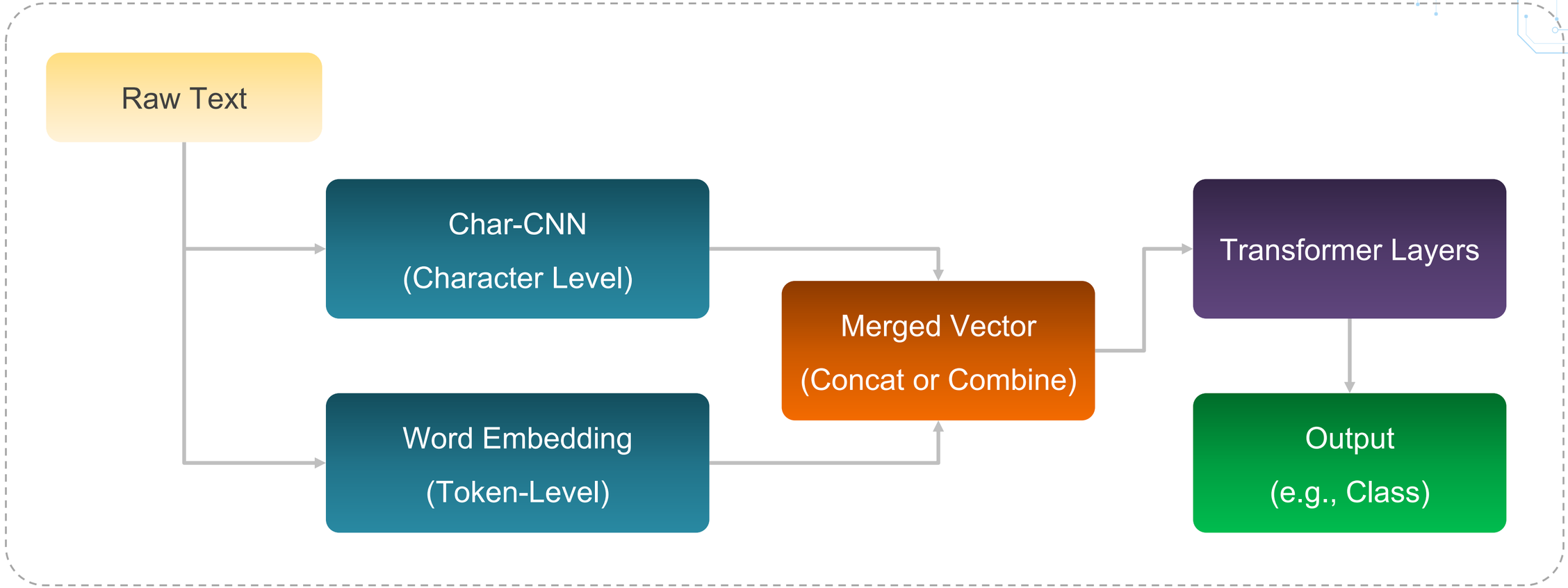
Combine character-level CNNs with Transformer models to improve performance.



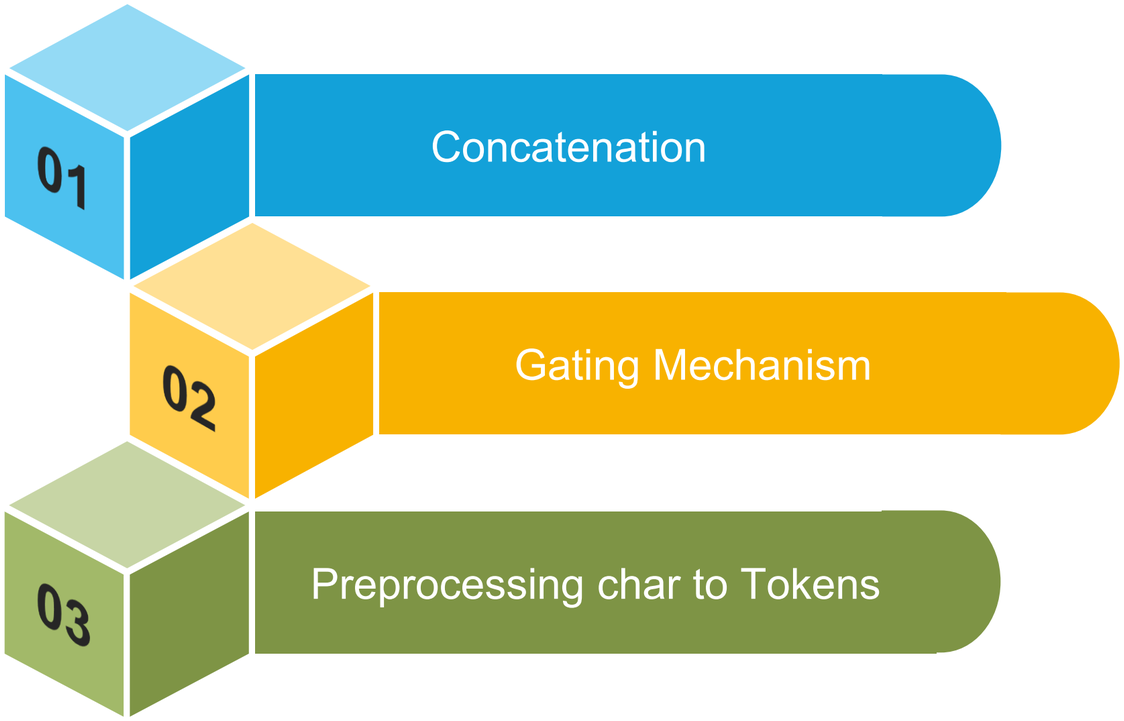
Why use Character-CNNs?

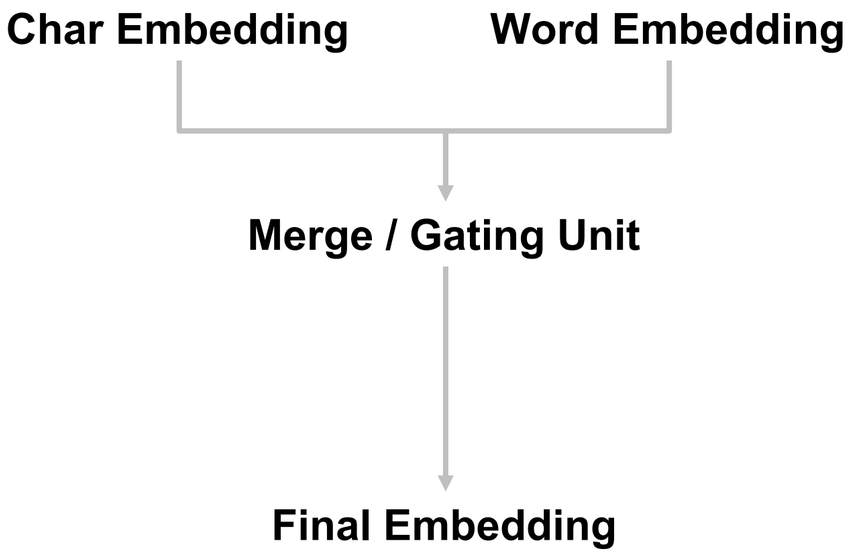


Model Architecture Overview



Fusion Strategies



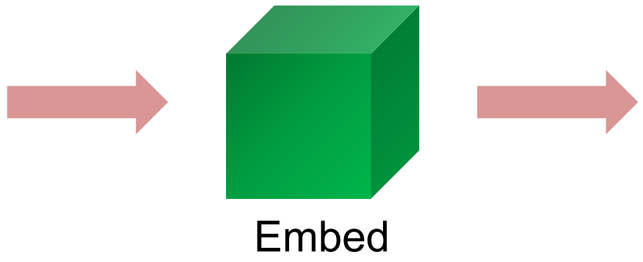


Applications of Character-Level Modeling in NLP Tasks

What are Sentence Embeddings?

Sentence BERT, also known as SBERT, is a transformative model designed for encoding sentence-level embeddings with a strong focus on semantic understanding.

- How old are you?
- What is your age?
- Where are you from?



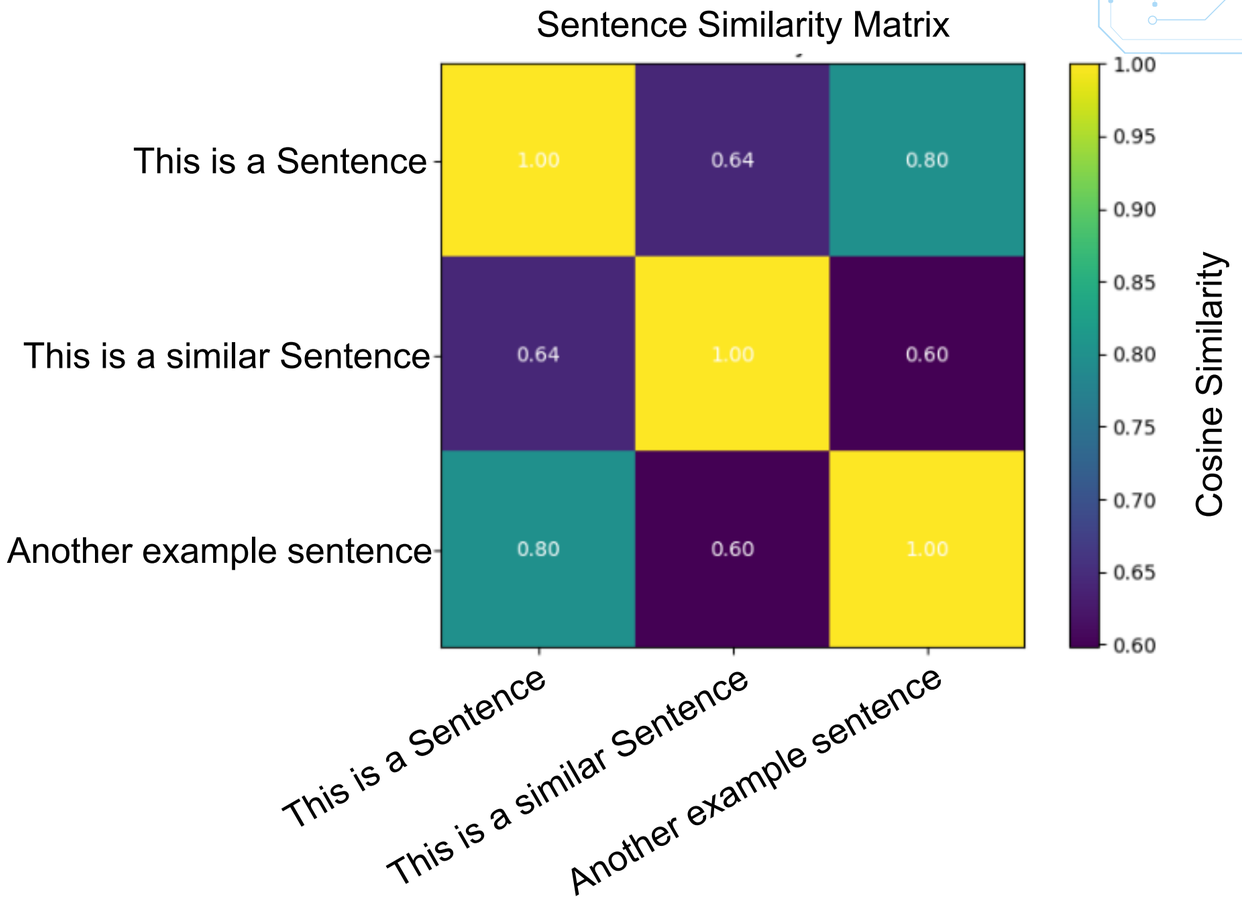
- [0.3, 0.2, …..]
- [0.2, 0.1, ….]
- [0.7, 0.6, …..]

Sentence-BERT and Universal Sentence Encoder

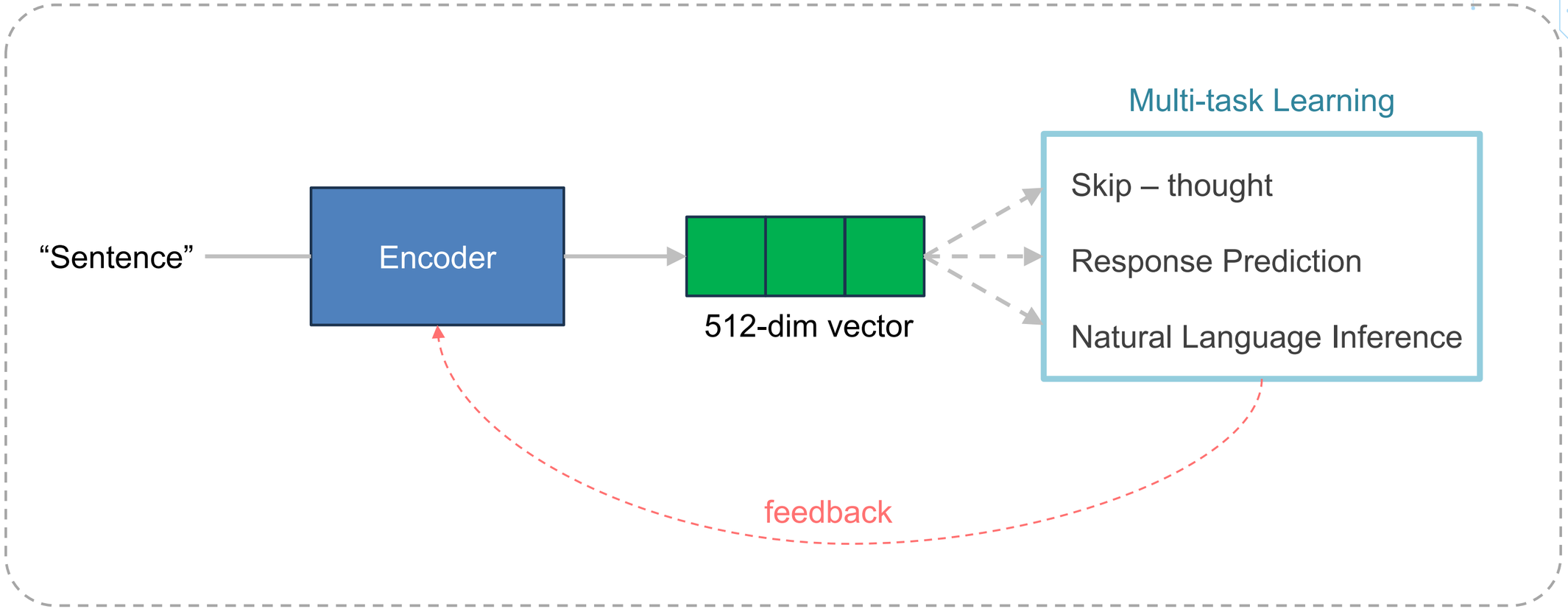
Sentence-Bert (SBERT)

SBERT adapts BERT to create fixed-size sentence embeddings, improving efficiency for sentence pair comparisons.

SBERT = Siamese / Triplet BERT + Pooling



Universal Sentence Encoder (USE)

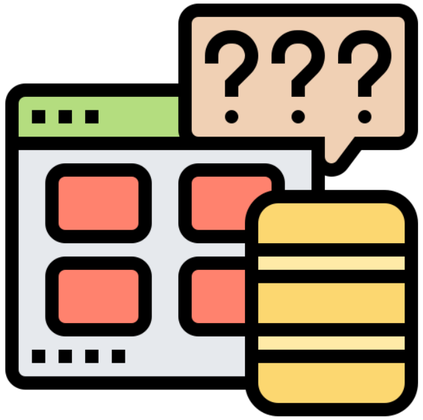


Text Clustering and Topic Modeling Using Sentence Embeddings

What is Text Clustering?

Clustering = Grouping similar texts without any labels

Organize customer reviews

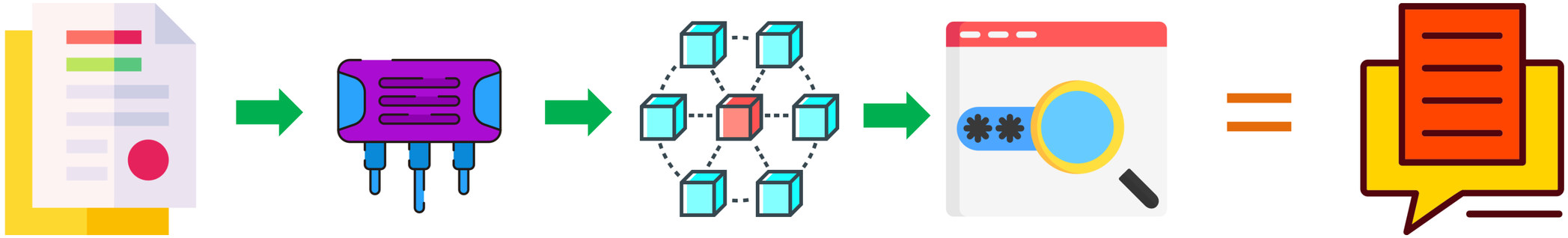


Group similar support tickets

Find duplicate questions

What is Topic Modelling?

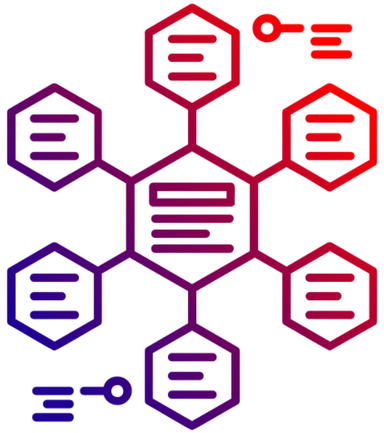
Topic Modeling = Extracting hidden topics from a large corpus of text



Docs → Cluster → Embeddings → Top keywords = Topic!

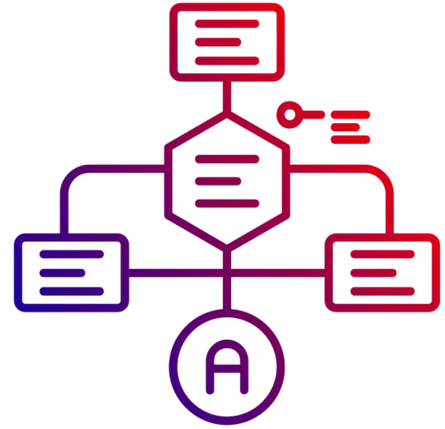
Why Clustering & Topic Modeling?

What if you have 10,000+ texts with no labels?



Group similar documents together

(Clustering)



Discover key themes or topics

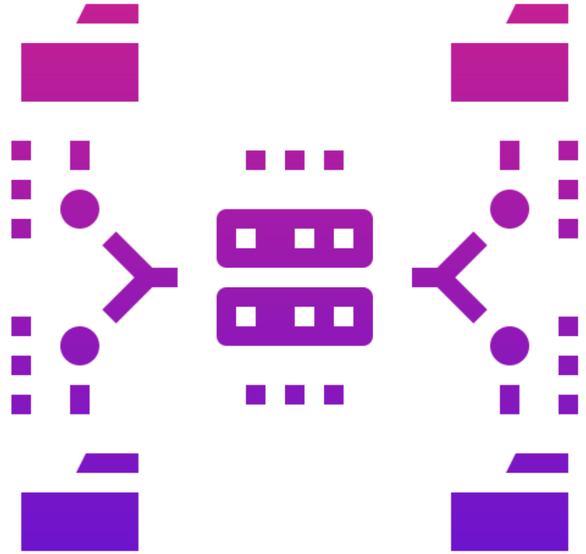
(Topic Modeling)

Modern Topic Modelling with Embeddings

DOCUMENTS

EMBEDDINGS

Cluster 1	Cluster 2	Cluster 3
machine learning	python	data cleaning
deep learning	coding	missing values
AI	programming	pandas



Subword Tokenization and Transformer Input Preparation (Demonstration)

Note: Refer to Module 4: Demo 1 on LMS for detailed steps.

Sentence Embeddings and Semantic Similarity in Hybrid Models (Demonstration)

Note: Refer to Module 4: Demo 2 on LMS for detailed steps.

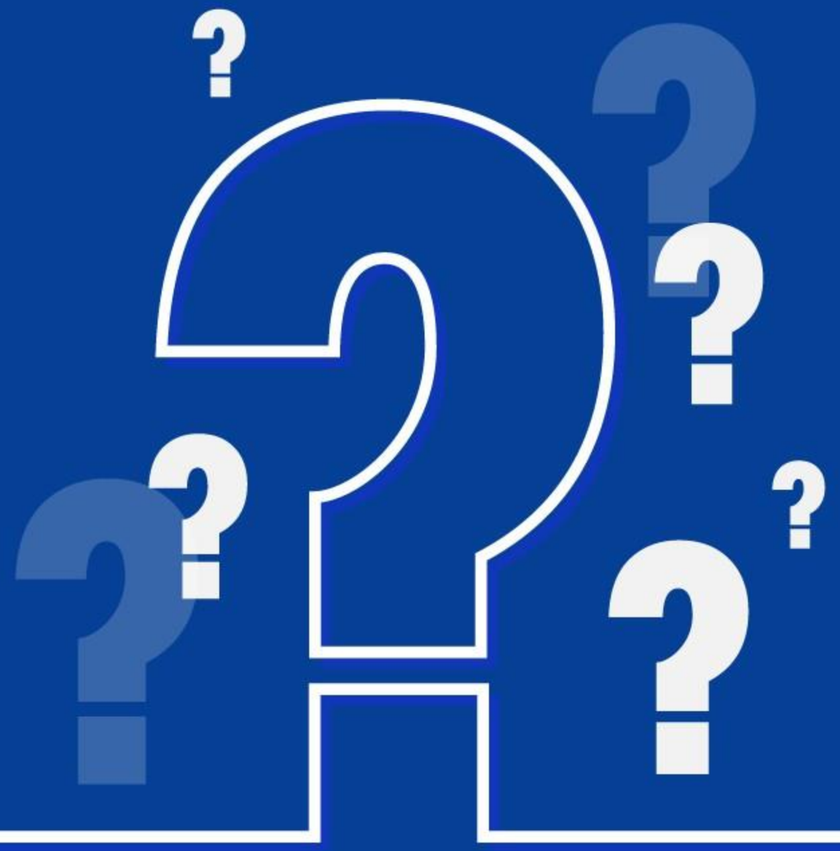
Summary

In this lesson, you have learned to:

- Perform subword tokenization using BPE, Unigram LM, and byte-level methods.
- Implementing WordPiece, SentencePiece, and N-gram-based tokenization techniques.
- Using tokenizers in transformer models like BERT and GPT for downstream tasks.
- Generating character and hybrid embeddings using CNNs, RNNs, and Transformers.
- Applying sentence embeddings for semantic search, chatbots, and clustering.

edureka! @ Veranda ENTERPRISE

Questions



edureka! a Veranda ENTERPRISE

Feedback

edureka! a Veranda ENTERPRISE

Thank You

For information, Please Visit our Website
www.edureka.co