7.One-Hot Encoding vs Bag of Words vs TF-IDF - Complete Comparison

Example Corpus

Document 1 (D_1): "the food is good"
Document 2 (D_2): "the food is bad"
Document 3 (D_3): "pizza is amazing"

Step 1: Building the Vocabulary

First, we extract all unique words from the corpus (ignoring case and duplicates):

Vocabulary = {the, food, is, good, bad, pizza, amazing}

Total unique words: **7**
Total documents: **3**

One-Hot Encoding

Theory

One-hot encoding represents each word as a binary vector where:

- The vector length equals the vocabulary size
- Only one position is "1" (hot), rest are "0" (cold)
- Each word gets a unique position in the vector

Vocabulary Index Mapping

Word	Index	One-Hot Vector
the	0	[1, 0, 0, 0, 0, 0, 0]
food	1	[0, 1, 0, 0, 0, 0, 0]
is	2	[0, 0, 1, 0, 0, 0, 0]
good	3	[0, 0, 0, 1, 0, 0, 0]
bad	4	[0, 0, 0, 0, 1, 0, 0]
pizza	5	[0, 0, 0, 0, 0, 1, 0]
amazing	6	[0, 0, 0, 0, 0, 0, 1]

Document Representation

Each document becomes a **sequence** of one-hot vectors:

D_1: "the food is good"

```
the     → [1, 0, 0, 0, 0, 0, 0]
food    → [0, 1, 0, 0, 0, 0, 0]
is      → [0, 0, 1, 0, 0, 0, 0]
good    → [0, 0, 0, 1, 0, 0, 0]
```

D_2: "the food is bad"

```
the     → [1, 0, 0, 0, 0, 0, 0]
food    → [0, 1, 0, 0, 0, 0, 0]
is      → [0, 0, 1, 0, 0, 0, 0]
bad     → [0, 0, 0, 0, 1, 0, 0]
```

D_3: "pizza is amazing"

```
pizza   → [0, 0, 0, 0, 0, 1, 0]
is      → [0, 0, 1, 0, 0, 0, 0]
amazing → [0, 0, 0, 0, 0, 0, 1]
```

Key Characteristics

- **Dimension:** Each word = 7-dimensional vector
- **Sparsity:** Extremely sparse (only 1 non-zero value)
- **No semantic meaning:** "good" and "bad" are equally distant
- **No frequency info:** Word repetition not captured
- **Multiple vectors per document**

Bag of Words (BoW)

Theory

Bag of Words represents each document as a frequency vector where:

- Each position corresponds to a word in the vocabulary
- The value indicates how many times the word appears
- Word order is ignored (hence "bag")
- **All words treated equally** (no importance weighting)

BoW Representation

Using the vocabulary: [the, food, is, good, bad, pizza, amazing]

Document	the	food	is	good	bad	pizza	amazing	Vector
D_1	1	1	1	1	0	0	0	[1, 1, 1, 1, 0, 0, 0]
D_2	1	1	1	0	1	0	0	[1, 1, 1, 0, 1, 0, 0]
D_3	0	0	1	0	0	1	1	[0, 0, 1, 0, 0, 1, 1]

Detailed Breakdown

D_1: "the food is good"

- "the" appears 1 time → position 0 = 1
- "food" appears 1 time → position 1 = 1
- "is" appears 1 time → position 2 = 1

- "good" appears 1 time → position 3 = 1
- Other words = 0

Key Limitation: The common word "is" (appears in all 3 documents) gets the same weight as distinctive words like "pizza" (appears in only 1 document).

Key Characteristics

- **Dimension:** Each document = 7-dimensional vector
- **One vector per document** (vs. multiple vectors in one-hot)
- **Captures frequency:** Can show if words repeat
- **No word order:** "food is good" = "good is food"
- **No word importance:** All words weighted equally
- **Fixed-length representation:** All documents same size vector

TF-IDF (Term Frequency-Inverse Document Frequency)

Theory

TF-IDF improves upon Bag of Words by weighting words based on their importance:

- **Term Frequency (TF):** How often a word appears in a document
- **Inverse Document Frequency (IDF):** How rare/distinctive the word is across all documents
- **Common words** (appearing everywhere) get lower weights
- **Distinctive words** (appearing in few documents) get higher weights

Step 1: Calculate Term Frequency (TF)

$$\text{TF}(word, document) = \frac{\text{Number of times word appears}}{\text{Total words in document}}$$

Document	the	food	is	good	bad	pizza	amazing
D_1 (4 words)	$\frac{1}{4}$ = 0.25	$\frac{1}{4}$ = 0.25	$\frac{1}{4}$ = 0.25	$\frac{1}{4}$ = 0.25	0	0	0

Document	the	food	is	good	bad	pizza	amazing
D_2 (4 words)	$\frac{1}{4}$ = 0.25	$\frac{1}{4}$ = 0.25	$\frac{1}{4}$ = 0.25	0	$\frac{1}{4}$ = 0.25	0	0
D_3 (3 words)	0	0	$\frac{1}{3}$ ≈ 0.33	0	0	$\frac{1}{3}$ ≈ 0.33	$\frac{1}{3}$ ≈ 0.33

Step 2: Calculate Inverse Document Frequency (IDF)

$$\text{IDF}(word) = \log_e\left(\frac{\text{Total documents} + 1}{\text{Documents containing word} + 1}\right) + 1$$

Word	Appears in	IDF Calculation	IDF Value
the	D_1, D_2 (2 docs)	$\log_e\left(\frac{3+1}{2+1}\right) + 1$	$\log_e(1.33) + 1$ ≈ **1.288**
food	D_1, D_2 (2 docs)	$\log_e\left(\frac{3+1}{2+1}\right) + 1$	$\log_e(1.33) + 1$ ≈ **1.288**
is	D_1, D_2, D_3 (3 docs)	$\log_e\left(\frac{3+1}{3+1}\right) + 1$	$\log_e(1) + 1$ = **1.000** ⭐
good	D_1 (1 doc)	$\log_e\left(\frac{3+1}{1+1}\right) + 1$	$\log_e(2) + 1$ ≈ **1.693**
bad	D_2 (1 doc)	$\log_e\left(\frac{3+1}{1+1}\right) + 1$	$\log_e(2) + 1$ ≈ **1.693**
pizza	D_3 (1 doc)	$\log_e\left(\frac{3+1}{1+1}\right) + 1$	$\log_e(2) + 1$ ≈ **1.693**
amazing	D_3 (1 doc)	$\log_e\left(\frac{3+1}{1+1}\right) + 1$	$\log_e(2) + 1$ ≈ **1.693**

Key Insight:

- "is" appears in ALL documents → lowest IDF (1.000) → less important
- "good", "bad", "pizza", "amazing" appear in only 1 document → highest IDF (1.693) → most important/distinctive

Step 3: Calculate TF-IDF Scores

$$\text{TF-IDF} = \text{TF} \times \text{IDF}$$

D_1: "the food is good"

Word	TF	×	IDF	=	TF-IDF
the	0.25	×	1.288	=	**0.322**

Word	TF	×	IDF	=	TF-IDF
food	0.25	×	1.288	=	**0.322**
is	0.25	×	1.000	=	**0.250** ⭐ (lowest)
good	0.25	×	1.693	=	**0.423** ⭐ (highest)
bad	0	×	1.693	=	0
pizza	0	×	1.693	=	0
amazing	0	×	1.693	=	0

Vector D_1: [0.322, 0.322, 0.250, 0.423, 0, 0, 0]

D_2: "the food is bad"

Word	TF	×	IDF	=	TF-IDF
the	0.25	×	1.288	=	**0.322**
food	0.25	×	1.288	=	**0.322**
is	0.25	×	1.000	=	**0.250** ⭐ (lowest)
good	0	×	1.693	=	0
bad	0.25	×	1.693	=	**0.423** ⭐ (highest)
pizza	0	×	1.693	=	0
amazing	0	×	1.693	=	0

Vector D_2: [0.322, 0.322, 0.250, 0, 0.423, 0, 0]

D_3: "pizza is amazing"

Word	TF	×	IDF	=	TF-IDF
the	0	×	1.288	=	0

Word	TF	×	IDF	=	TF-IDF
food	0	×	1.288	=	0
is	0.33	×	1.000	=	**0.330** ⭐ (lowest)
good	0	×	1.693	=	0
bad	0	×	1.693	=	0
pizza	0.33	×	1.693	=	**0.559** ⭐ (highest)
amazing	0.33	×	1.693	=	**0.559** ⭐ (highest)

Vector D_3: [0, 0, 0.330, 0, 0, 0.559, 0.559]

TF-IDF Summary Table

Document	the	food	is	good	bad	pizza	amazing	Vector
D_1	0.322	0.322	0.250	**0.423**	0	0	0	[0.322, 0.322, 0.250, **0.423**, 0, 0, 0]
D_2	0.322	0.322	0.250	0	**0.423**	0	0	[0.322, 0.322, 0.250, 0, **0.423**, 0, 0]
D_3	0	0	0.330	0	0	**0.559**	**0.559**	[0, 0, 0.330, 0, 0, **0.559**, **0.559**]

Key Characteristics

- **Dimension:** Each document = 7-dimensional vector
- **One vector per document**
- **Captures frequency AND importance**
- **Common words automatically de-emphasized** (lower values)
- **Distinctive words emphasized** (higher values)
- **Better for classification** than BoW

Complete Comparison

Aspect	One-Hot Encoding	Bag of Words	TF-IDF
Unit	Individual words	Entire document	Entire document
Output	Multiple vectors per document	Single vector per document	Single vector per document
Frequency	No (binary only)	Yes (raw counts)	Yes (weighted counts)
Word Importance	No	No (all equal)	**Yes (weighted)** ⭐
Common Words	Same as rare words	Same as rare words	**De-emphasized** ⭐
Distinctive Words	Same as common	Same as common	**Emphasized** ⭐
Dimension	Vocabulary size	Vocabulary size	Vocabulary size
Sparsity	Extremely sparse	Moderately sparse	Moderately sparse
Use case	Neural network inputs	Basic classification	**Better classification** ⭐
Semantic meaning	None	None	None
Performance	Baseline	Good	**Better than BoW** ⭐

Example with Repetition

If we had: **D_4: "good good good food"**

One-Hot Encoding

Still just individual vectors (good appears 3 times as separate vectors):

```
good → [0, 0, 0, 1, 0, 0, 0]
good → [0, 0, 0, 1, 0, 0, 0]
good → [0, 0, 0, 1, 0, 0, 0]
food → [0, 1, 0, 0, 0, 0, 0]
```

Bag of Words

Single vector capturing raw frequency:

```
D₄ → [0, 1, 0, 3, 0, 0, 0]
         ↑       ↑
        food    good (3 times)
```

All words weighted equally (no importance consideration).

TF-IDF

Single vector with weighted frequency:

TF Calculation:

- good: $\frac{3}{4}$ = 0.75
- food: $\frac{1}{4}$ = 0.25

TF-IDF Calculation:

- good: 0.75×1.693 ≈ **1.270** (high frequency × high importance)
- food: 0.25×1.288 ≈ **0.322**

```
D₄ → [0, 0.322, 0, 1.270, 0, 0, 0]
         ↑           ↑
        food    good (weighted by importance AND frequency)
```

Key Insight: TF-IDF captures both:

1. How often "good" appears (frequency = 3 times)
2. How important "good" is (appears in only 1 document, so distinctive)

Visual Comparison Summary

```
One-Hot:  Word-level  | No frequency | No importance | Multiple vectors
            ↓
Bag of Words: Document-level | Counts frequency | No importance | Single vector
            ↓
TF-IDF:    Document-level | Counts frequency | Weights importance | Single vector ⭐
```

When to Use Each Method?

Method	Best Use Case
One-Hot	Neural networks (embedding layers), categorical features
Bag of Words	Simple baselines, when word importance doesn't matter
TF-IDF	Text classification, document similarity, when word importance matters ⭐

Conclusion

TF-IDF is superior to Bag of Words because it automatically identifies and emphasizes important words while de-emphasizing common words. This leads to better machine learning model performance in most text classification and retrieval tasks.

However, all three methods share limitations:

- No semantic understanding (synonyms treated differently)
- Sparse representations
- Out-of-vocabulary problems

Modern approaches like Word2Vec, GloVe, and transformer embeddings (BERT) address these limitations.