Text Representation

Lecture 2: The Bag-of-Words Model

Harito

September 15, 2025

From Text to Numbers

The Challenge

Machine learning algorithms (like classifiers, regressors) operate on numerical data, not raw text.

From Text to Numbers

The Challenge

Machine learning algorithms (like classifiers, regressors) operate on numerical data, not raw text.

How can we convert a sentence like ‘”I love NLP”‘ into a vector of numbers?

This process is called **Text Representation** or **Feature Extraction**.

The Bag-of-Words (BoW) Model

The simplest and one of the most common models is Bag-of-Words.

- It describes the occurrence of each word within a document.
- It disregards grammar and even word order.
- It only cares about which words appear and their frequency.
- The "bag" analogy: imagine putting all tokens from the document into a bag and counting them.

Count Vectorization: The BoW Recipe

How to implement the BoW model?

1. **Collect Corpus**: Gather all your documents.
2. **Tokenize**: Use a tokenizer (from Lab 1!) to break each document into tokens.
3. **Build Vocabulary**: Collect all unique tokens from the entire corpus and assign a unique integer index to each one.
4. **Create Vectors**: For each document, create a vector. The vector's length is the size of the vocabulary. For each token in the document, increment the count at its corresponding index in the vector.

Step-by-Step Example

1. **Corpus:**
   - D1: "I love NLP."
   - D2: "I love programming."

Step-by-Step Example

1. **Corpus:**
   - D1: "I love NLP."
   - D2: "I love programming."
2. **Tokenize (lowercase, ignore punctuation):**
   - D1 tokens: ‘['i', 'love', 'nlp']‘
   - D2 tokens: ‘['i', 'love', 'programming']‘

Step-by-Step Example

1. **Corpus:**
   - D1: "I love NLP."
   - D2: "I love programming."
2. **Tokenize (lowercase, ignore punctuation):**
   - D1 tokens: ‘['i', 'love', 'nlp']‘
   - D2 tokens: ‘['i', 'love', 'programming']‘
3. **Build Vocabulary:**
   - Unique tokens: ‘'i', 'love', 'nlp', 'programming'‘
   - Vocabulary Map: ‘'i': 0, 'love': 1, 'nlp': 2, 'programming': 3‘

Step-by-Step Example (Cont.)

Vocabulary: ‘’i’: 0, ’love’: 1, ’nlp’: 2, ’programming’: 3‘

4. Create Vectors (Vector length = 4):

- **D1: ”I love NLP”** -> ‘[’i’, ’love’, ’nlp’]‘

i	love	nlp	programming
1	1	1	0

Vector: ‘[1, 1, 1, 0]‘

Step-by-Step Example (Cont.)

Vocabulary: ‘’i’: 0, ’love’: 1, ’nlp’: 2, ’programming’: 3‘

4. Create Vectors (Vector length = 4):

- **D1: ”I love NLP”** -> ‘[’i’, ’love’, ’nlp’]‘

i	love	nlp	programming
1	1	1	0

Vector: ‘[1, 1, 1, 0]‘

- **D2: ”I love programming”** -> ‘[’i’, ’love’, ’programming’]‘

i	love	nlp	programming
1	1	0	1

Vector: ‘[1, 1, 0, 1]‘

The Document-Term Matrix

The final output is a matrix where rows are documents and columns are vocabulary terms.

	i	love	nlp	programming
Document 1	1	1	1	0
Document 2	1	1	0	1

This matrix is the numerical representation of our corpus, ready for a machine learning model!

Limitations of BoW

While powerful, the Bag-of-Words model has limitations:

- **Vocabulary Size**: The vocabulary can become huge for large corpora, leading to very long (sparse) vectors.
- **Sparsity**: Most vectors will be filled with zeros, which can be inefficient.
- **No Word Order**: ‘”John likes Mary”‘ and ‘”Mary likes John”‘ have the exact same representation. Context is lost.
- **No Semantic Meaning**: It doesn’t know that ‘”good”‘ and ‘”great”‘ are similar. It just sees two different words.

Next Steps

Time for Lab 2!

Objective:

- Implement a ‘Vectorizer‘ interface.
- Create your own ‘CountVectorizer‘ that uses your ‘Tokenizer‘.