UNIVERSIDADE CATÓLICA PORTUGUESA

BRAGA

Behavior Analysis Technologies

Session 3

Representing Text

Applied Data Science

2024/2025

Question

How can text be encoded for machine interpretation?

Representing Text

- How to efficiently represent text for machine processing?
  - To measure the similarity between texts (**text similarity**);
  - To categorize documents (**text classification**);
  - To group documents together (**text clustering**);
  - To decide if a document is relevant to a search query (**search**).

- We need **methematical representations** for words (aka **terms**) and for sequences of words (aka **documents**).

Representing Words

- **Discrete (sparse) representation:**
  - Each word is assigned a unique **term ID** (an integer).
  - The **vocabulary** V consists of a **mapping between words (terms) and their corresponding IDs**.

Vocabulary

ID	Term
1	each
2	word
3	assigned
4	unique
5	term
...	...

Representing Words

- **Continuous (dense) representation:**

  - Each word is represented as a real-valued vector within a lower-dimensional latent space, known as an **embedding vector**.

  - Words with similar or related meanings are positioned close to each other within this embedding space.

  - We will cover this later!

$$\vec{w} = \begin{array}{|c|c|c|c|c|c|} \hline 0.15 & 0.07 & 0.83 & 0.46 & \ldots & 0.02 \\ \hline \end{array}$$

Representing Documents

- Words are represented by their respective IDs in the vocabulary.
  - Some words may be **out of vocabulary** (OOV).
  - OOV words can be **replaced** by a special OOV token (if word position matters) or simply **ignored**.

- A document can be represented as a **sequence of words** or as a **term vector**:
  - In the term vector, each element corresponds to a word in the vocabulary.
  - The value of each element can indicate:
    - The **presence or absence** of a word (binary: 0/1)
    - The **frequency** of the word (integer)
    - The **importance** of the word (real-valued score).

Representing Documents

UNIVERSIDADE CATÓLICA PORTUGUESA BRAGA

Vocabulary

ID	Term
1	each
2	word
3	assigned
4	unique
5	term
...	...

$$\mathrm{d} = \text{"each word is unique"}$$

$$d = [t1, t2, OOV, t4]$$

$$\vec{d} = \langle 1, 1, 0, 1, 0, \ldots \rangle$$

Representing a Collection of Documents

- Document-term matrix, where:
  - Rows correspond to documents
  - Columns correspond to terms in the vocabulary

- Generally, the obtained matrix is huge, but most of the values are zeros (sparse matrix).

	t_1	t_2	t_3	$\dots$	t_m
d_1	1	0	2		0
d_2	0	1	0		2
d_3	0	0	1		0
$\dots$					
d_n	0	1	0		0

Document-term matrix

One-Hot-Encoding

- Each word in the vocabulary is represented as **a vector where all elements are zero except one** (corresponding to the index of the word).
- **Very sparse**, **high-dimensional**, and lacks information about semantic relationships between words.

id	color
1	green
2	blue
3	red
4	red
5	green

id	green	blue	red
1	1	0	0
2	0	1	0
3	?	?	?
4	?	?	?
5	?	?	?

One-Hot-Encoding

- Each word in the vocabulary is represented as **a vector where all elements are zero except one** (corresponding to the index of the word).
- **Very sparse**, **high-dimensional**, and lacks information about semantic relationships between words.

id	color
1	green
2	blue
3	red
4	red
5	green

id	green	blue	red
1	1	0	0
2	0	1	0
3	0	0	1
4	0	0	1
5	1	0	0

One-Hot-Encoding

- **The Dummy Variable Trap:**
  - Occurs when two or more dummy variables created by one-hot encoding are highly correlated (**multi-collinear**);
  - This means that **one variable can be predicted from the others**.

id	color
1	green
2	blue
3	red
4	red
5	green

→

id	green	blue
1	1	0
2	0	1
3	0	0
4	0	0
5	1	0

Exercise: One-Hot-Encoding

- Follow the instrunction in: https://github.com/LCDA-UCP/tac-hands-on

- **Exercise:** Implement One-Hot-Encoding with Python (using only the numpy library)
    - Use the one_hot_encoding.py script as base

- **Take-home assignment:** Add support for dealing with the dummy variable trap

Bag of Words

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- A simple, commonly used text representation method.
- Represents a document by **counting the occurrences of each word**, disregarding grammar and word order.
- **How it works?**
  1. **Tokenization:** Split text into individual words (tokens).
  2. **Vocabulary Creation:** Build a list of unique words (vocabulary) from the entire corpus.
  3. **Vector Representation:** Each document is represented as a vector of word counts.

Bag of Words

- Let's consider 3 documents (sentences):
  - The cat in the hat.
  - The dog in the house.
  - The bird in the sky.

Bag of Words

- **Tokenization**
  - The cat in the hat.
  - The dog in the house.
  - The bird in the sky.

The	cat	in	the	hat
The	dog	in	the	house
The	bird	in	the	sky

Bag of Words

- **Vocabulary Creation**
  - The cat in the hat.
  - The dog in the house.
  - The bird in the sky.

Vocabulary

The	cat	in	hat	dog	house	bird	sky

Bag of Words

- **Vector Representation**

Text	the	cat	in	hat	dog	house	bird	sky
The cat in the hat.	2	1	1	1	0	0	0	0
The dog in the house.	2	0	1	0	1	1	0	0
The bird in the sky.	2	0	1	0	0	0	1	1

Exercise: Bag of Words

- **Exercise:** Implement Bag of Words with Python (using only the numpy library)
  - Use the bag_of_words.py script as base

- **Take-home assignment:** explore scikit-learn and find out how can you create a bag of words with it.

N-gram

- A **contiguous sequence of N words** in a text.

- Captures some **word order and contextual relationships** between words (unlike Bag of Words).

- **Unigrams (N=1)**: Single words (["I", "love", "my", "dog"]).
- **Bigrams (N=2)**: Pairs of consecutive words (["I love", "love my", "my dog"]).
- **Trigrams (N=3)**: Triplets of consecutive words (["I love my", "love my dog"]).

Exercise: N-gram

- **Exercise:** Implement n-gram with Python
    - Use the n_grams.py script as base

- **Take-home assignment:** Extend the NGram class to include some preprocessing like converting text to lowercase, removing punctuation and stop words.

TF-IDF

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- TF-IDF (Term Frequency - Inverse Document Frequency) is a numerical statistic that reflects the **importance of a word within a document relative to a collection of documents**, known as a **corpus**.

$$\text{TF-IDF}(t, d) = TF(t, d) * IDF(t)$$

- **Term Frequency** (TF)
    - Measures the frequency of a term (t) within a document (d).

$$TF(t, d) = \frac{\text{number of times } t \text{ appears in d}}{\text{total number of terms in } d}$$

TF-IDF

- TF-IDF (Term Frequency - Inverse Document Frequency) is a numerical statistic that reflects the **importance of a word within a document relative to a collection of documents**, known as a **corpus**.

$$\text{TF-IDF}(t,d) = TF(t,d) * IDF(t)$$

- **Inverse Document Frequency** (IDF)
  - Measures the rarity of a term across a collection of documents.

$$IDF(t,D) = \log\left(\frac{\text{Total number of documents in the corpus } N}{\text{Number of documents containing term } t}\right)$$

TF-IDF

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- TF-IDF (Term Frequency - Inverse Document Frequency) is a numerical statistic that reflects the **importance of a word within a document relative to a collection of documents**, known as a **corpus**.

$$\text{TF-IDF}(t,d) = TF(t,d) * IDF(t)$$

$$\text{TF-IDF}(t,d) = \frac{\text{number of times } t \text{ appears in d}}{\text{total number of terms in } d} * \log\left(\frac{\text{Total number of documents in the corpus } N}{\text{Number of documents containing term } t}\right)$$

- **Note:** some libraries including sklearn use a slightly different formula for the IDF

$$IDF(t,d) = \log\left(\frac{\text{Total number of documents in the corpus } N}{\text{Number of documents containing term } t}\right) + 1$$

Exercide: TF-IDF

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- **Exercise:** Manualy calculate the TF-IDF

- **Exercise:** Implement TF-IDF with Python
    - Use the tf_idf.py script as base

- **Take-home assignment:** Extend the TFIDF class to include some preprocessing like converting text to lowercase, removing punctuation and stop words.

Vector Embedding of Words

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- **To be covered in detail later!**

- Mapping a word to a **vector**:
  - The semantic of the word is embedded in the vector.

- Word embeddings depeng on **word similarity**.
  - Similar words occur in similar contexts – they are exchangable.

- Transfer learning for text.

Word Embeddings

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- Stores each word in as **a point in space**, where it is represented by a **dense vector** of fixed number of dimensions.
  - For example, "cat" might be represented as: , [0.4, -0.11, 0.55, ..., 0.1, 0.002]
- **Unsupervised**, built by processing large corpus.
- Assumes **dependence between words**.