Векторизация текста (Text Data Vectorization)

> Разделы: Алгоритмы

Векторизация текста — это процесс конвертации текста в числа. Следует помнить о том, что компьютер не способен обрабатывать слова — только числа, поэтому возникает необходимость в таком преобразовании для обеспечения корректной работы алгоритмов машинного обучения.

Начальным этапом векторизации текста является его разбиение на элементы:

- **Токены** — текстовые единицы, т.е. слова, словосочетания, предложения и т.п. Чаще всего в качестве токена выбирается слово. Множество токенов образуют *словарь*.
- **Документы** — множества токенов, принадлежащих одной смысловой единице. В качестве документа может выступать предложение или небольшой текст.
- **Корпус** — множество всех документов.

Таким образом, в процессе векторизации, текст преобразуется в массив — корпус, состоящий из массивов меньшего размера — документов, а элементами документов являются вектора — токены.

Можно выделить несколько методов векторизации текста. Самым простым из всех является **прямое кодирование**. В случае прямого кодирования получается массив, строки которого представляют собой токены корпуса, а столбцы — положение токена в документе. Полученная матрица состоит из двух элементов: 0 и 1, где единица соответствует вхождению слова в документ, а 0 — отсутствию слова в документе.

Приведем пример. Рассмотрим четверостишие из поэмы «Руслан и Людмила»:

> У Лукоморья дуб зеленый;
> Златая цепь на дубе том;
> И днем, и ночью кот ученый.
> Все ходит по цепи кругом.

Будем считать две строфы четверостишия отдельным документом. Для того, чтобы составить словарь токенов данного текста, потребуется привести все слова к инфинитиву и расположить их по алфавиту: «все», «день», «дуб», «зеленый», «златой», «и», «кот», «кругом», «лукоморье», «на», «ночь», «по», «том», «у», «ученый», «ходит», «цепь».

Как правило, несамостоятельные части речи, такие как местоимения, предлоги и союзы, отбрасываются при составлении словаря, однако здесь целесообразно их сохранить в силу компактности текста. Теперь конвертируем текст в числа. Первая строфа содержит 4 слова, вторая — 5 слов. Вектор этого документа будет иметь вид.

Номер слова в документе	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
все	0	0	0	0	0	0	0	0	0
день	0	0	0	0	0	0	0	0	0
дуб	0	0	1	0	0	0	0	1	0
зеленый	0	0	0	1	0	0	0	0	0
златой	0	0	0	0	1	0	0	0	0
и	0	0	0	0	0	0	0	0	0
кот	0	0	0	0	0	0	0	0	0
кругом	0	0	0	0	0	0	0	0	0
лукоморье	0	1	0	0	0	0	0	0	0
на	0	0	0	0	0	0	1	0	0
ночь	0	0	0	0	0	0	0	0	0
по	0	0	0	0	0	0	0	0	0
том	0	0	0	0	0	0	0	0	1
у	1	0	0	0	0	0	0	0	0
ученый	0	0	0	0	0	0	0	0	0
ходит	0	0	0	0	0	0	0	0	0
цепь	0	0	0	0	0	1	0	0	0

Аналогичным образом закодируем второй документ (третью и четвертую строфу) нашего стихотворения. Получим следующий массив.

Номер слова в документе	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
все	0	0	0	0	0	0	1	0	0	0	0

Номер слова в документе	1	2	3	4	5	6	7	8	9	10	11
день	0	1	0	0	0	0	0	0	0	0	0
дуб	0	0	0	0	0	0	0	0	0	0	0
зеленый	0	0	0	0	0	0	0	0	0	0	0
златой	0	0	0	0	0	0	0	0	0	0	0
и	1	0	1	0	0	0	0	0	0	0	0
кот	0	0	0	0	1	0	0	0	0	0	0
кругом	0	0	0	0	0	0	0	0	0	0	1
лукоморье	0	0	0	0	0	0	0	0	0	0	0
на	0	0	0	0	0	0	1	0	0	0	0
ночь	0	0	0	1	0	0	0	0	0	0	0
по	0	0	0	0	0	0	0	0	1	0	0
том	0	0	0	0	0	0	0	0	0	0	0
у	0	0	0	0	0	0	0	0	0	0	0
ученый	0	0	0	0	0	1	0	0	0	0	0
ходит	0	0	0	0	0	0	0	1	0	0	0
цепь	0	0	0	0	0	0	0	0	0	1	0

Таким образом, нами был закодирован небольшой текст с помощью алгоритма **прямого кодирования**.

Методы, предполагающие бинарное кодирование, позволяют обрабатывать текст быстрее, чем те, которые основаны на работе с частотой встречаемости слова или его контекстом. При этом оно характеризуются небольшой величиной потерь, однако игнорирует многие свойства языка. Например, прямое кодирование создает громоздкие массивы даже из небольших текстов (как в рассмотренном примере), а другой распростаненный метод — Bag of words не рассматривает положение слова в документе.

На данный момент популярными методами векторизации текста являются TF-IDF и Word embeddings. Word embeddings — это векторное представление слова. Для вектора вводится размерность, состоящая из смысловой нагрузки слов в тексте. Словам присваиваются числа — ранги слова в подходящем контексте.

Затем при помощи методики *Word2vec* реализуется конечное векторное представление слов и предсказание на основании него. Архитектура Word2vec глобально подразделяется на два вида — *Skip-gram* и *Continuous bad of words*.

Архитектура вида Skip-gram получает на вход слово и пытается определить для него контекст. Continuous bad of words, в свою очередь, пытается предсказать слово, исходя из его контекста. Таким образом, Word2vec представляет собой мощный инструмент для анализа текстовых данных, который учитывает особенности языка и контекст, в котором используется слово.