Машинное обучение и работа с текстами

ФиПЛ-2022

Где у текстов признаки?

- Слова в тексте – это наши признаки. Но нужно понять, как их представить в числовом виде
- Превращаем слова в числа?

Где у текстов признаки?

- Слова в тексте – это наши признаки. Но нужно понять, как их представить в числовом виде
- Превращаем слова в числа:
- Можно использовать One Hot Encoding:

	1	2	3	4	5	6	7	8
I	1	0	0	0	0	0	0	0
ate	0	1	0	0	0	0	0	0
an	0	0	1	0	0	0	0	0
apple	0	0	0	1	0	0	0	0
and	0	0	0	0	1	0	0	0
played	0	0	0	0	0	1	0	0
the	0	0	0	0	0	0	1	0
piano	0	0	0	0	0	0	0	1

- в чем проблема таких векторов?

Где у текстов признаки?

- Текст – сам по себе признак. Но нужно понять, как его представить в числовом виде
- Превращаем слова в числа:
- Можно использовать One Hot Encoding:

	1	2	3	4	5	6	7	8
I	1	0	0	0	0	0	0	0
ate	0	1	0	0	0	0	0	0
an	0	0	1	0	0	0	0	0
apple	0	0	0	1	0	0	0	0
and	0	0	0	0	1	0	0	0
played	0	0	0	0	0	1	0	0
the	0	0	0	0	0	0	1	0
piano	0	0	0	0	0	0	0	1

- это очень неэффективно и вообще ни о чем нам не говорит!

Идея: Bag of Words

- Составляем словарь для всех наших текстов, сортируем и нумеруем слова
- Каждый текст – это вектор такой же длины, какой у нас словарь
- Вписываем частоту слова под его порядковым номером

	she	loves	pizza	is	delicious	a	good	person	people	are	the	best
She loves pizza, pizza is delicious	1	1	2	1	1	0	0	0	0	0	0	0
She is a good person	1	0	0	1	0	1	1	1	0	0	0	0
good people are the best	0	0	0	0	0	0	1	0	1	1	1	1

Идея: Bag of Words

- Это уже неплохая идея: мы получаем представление как минимум о частоте слова, поэтому если в нашем тексте 10 раз встретилось «купи», вероятно, это спам.
- В sk-learn эта идея реализована в инструменте для предобработки признаков CountVectorizer().
- Можно эту идею улучшить!
- Почему бы вместо слов не использовать n-грамы?

А что такое n-грамы?

- **”This is not good at all”** =
  - “This is”
  - “is not”
  - “not good”
  - “good at”
  - “at all”

BoW + n-grams

- Если в нашем словаре будут n-грамы, мы будем учитывать не только частоту слов, но и их сочетаемость!
- Какой недостаток у этого подхода?

BoW + n-grams

- Если в нашем словаре будут n-грамы, мы будем учитывать не только частоту слов, но и их сочетаемость!
- Биграмов будет больше, чем униграмов; триграмов будет еще больше; ну и так далее...
- => У нас будут очень длинные вектора!
- Тем не менее, иногда контекст учитывать важнее.

TF-IDF

- Давайте еще усовершенствуем нашу модель.
- Как вам частоты словечек типа «и», «или», «в»?

TF-IDF

- Давайте еще усовершенствуем нашу модель.
- Наверное, хочется как-то учитывать еще и «важность» слов: чисто интуитивно слово «котик» важнее, чем слово «и».
- Как это сделать?

TF-IDF

- Давайте еще усовершенствуем нашу модель.
- Наверное, хочется как-то учитывать еще и «важность» слов: чисто интуитивно слово «котик» важнее, чем слово «и».
- Давайте вместо частоты будем записывать term frequency-inverse document frequency.
- Term frequency (частота слова) – отношение частоты слова к общему числу всех слов
- IDF (обратная частота документа) – инверсия частоты, с которой слово встречается в разных документах

TF-IDF explained

- TF: $\frac{\text{частота слова}}{\text{все слова в корпусе}}$
- IDF: $\log \frac{\text{количество документов в корпусе}}{\text{количество документов,в которых встречается наше слово}}$
- Пример:

  у нас в корпусе 3 документа, в которых в сумме 100 слов.
  слово «котик» встречается 30 раз, но в двух документах.
  его TF-IDF = $\frac{30}{100} \cdot \log \frac{3}{2}$ = 0.12
  а слово «карбюратор» встречается 15 раз, но только в одном документе.
  его TF-IDF = $\frac{15}{100} \cdot \log \frac{3}{1}$ = 0.16

TF-IDF explained

- Есть, однако, одна засада...
- Посчитайте TF-IDF для слова «и», которое в нашем корпусе из 3 документов и 100 слов встретилось 40 раз и во всех трех документах.
- *Подсказка: log(1) = 0*

TF-IDF explained

- Получается, что TF-IDF для такого слова будет равно 0
- Но и для слова, которого в наших документах вообще нет, тоже 0
- Поэтому обычно используют *сглаживание* и таким словам приписывают какое-нибудь очень маленькое значение.

- TF-IDF реализован в sk-learn как TfidfVectorizer().