Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем

Клочков Владислав Сергеевич, 342

# Машинное обучение в задачах классификации текстов

Отчёт по учебной практике

Научный руководитель:
к. ф-м. н. доцент Графеева Н.Г.

Санкт-Петербург
2022

# Оглавление

# 1. Введение

В современном мире становится все больше информации. И появляется необходимость работать с большим количеством данных. Если обрабатывать всю информацию вручную, то на это уйдет колоссальное количество времени. Поэтому появляется потребность в автоматизации такой работы. Здесь в дело вступает машинное обучение. Машинное обучение — это математические, статистические и вычислительные методы для разработки алгоритмов, способных решить задачу не прямым способом, а на основе поиска закономерностей во входных данных.

Классификация — это один из разделов машинного обучения. Задача классификации — предсказание категории обьекта согласно определенным заданным заранее признакам.

Отдельное направление классификации является NLP(natural language processing) [1]. NLP — пересечение машинного обучения и математической лингвистики, направленное на изучение методов анализа и синтеза естественного языка.

# 2. Цель и задачи

Цель: разработать и применить модель для классификации текста.

Для достижения этой цели были поставлены следующие задачи:

- изучить и применить алгоритм предобработки текста;
- изучить и применить модели классификации текста;
- оценить результаты.

# 3. Предобработка текста

Прежде, чем обучать модель, необходимо подготовить текст. А именно:

- Токенизация
- Приведение к нижнему регистру, удаление знаков препинания и стоп-слов
- Лемматизация
- Стемминг

Возьмем в качестве примера текст из датасета и посмотрим поближе, что делает каждый шаг.

```
«Победа» задумалась над введением платы за пластиковые стаканчики на борту
```

Рис. 1: Начальный текст

## 3.1. Токенизация

Это разбиение текста на отдельные слова, знаки препинания, границы абзацев и т.п.

```
['«', 'Победа', '»', 'задумалась', 'над', 'введением', 'платы', 'за', 'пластиковые', 'стаканчики', 'на', 'борту']
```

Рис. 2: Текст после токенизации

## 3.2. Приведение к нижнему регистру, удаление знаков препинания и стоп-слов

Приводим слова к нижнему регистру, потому что «Победа» и «победа» считается как два разных слова. Убираем знаки препинания, так как какие-то зависимости по частоте их использования найти довольно сложно. Удаляем стоп-слова, то есть слова, который не несут какой-то смысловой нагрузки. Например, предлоги, союзы, местоимения и т.д.

```
['победа', 'задумалась', 'введением', 'платы', 'пластиковые', 'стаканчики', 'борту']
```

Рис. 3: Текст в нижнем регистре после удаления знаков препинания и стоп-слов

## 3.3. Лемматизация

Это метод морфологического анализа, который приводит слово к первоначальной форме. Для существительных — это именительный падеж, единственное число. Для глаголов — инфинитив. Для прилагательных — именительный падеж, единственное число, мужской род.

```
['победа', 'задуматься', 'введение', 'плата', 'пластиковый', 'стаканчик', 'борт']
```

Рис. 4: Текст после лемматизации

## 3.4. Стемминг

Это процесс приведения слова к его основе. Другими словами отсечение корней и суффиксов слов.

```
['побед', 'задума', 'введен', 'плат', 'пластиков', 'стаканчик', 'борт']
```

Рис. 5: Текст после стемминга

# 4. Векторизация

Следующий шаг после предобработки текста является векторизация. Это процесс конвертации текста в числа для дальнейшего использования в алгоримах.

Одна из расспространенных методик — это «Bag of words» или «Мешок слов» [2]. В логике этой методики два предложения считаются одинаковыми, если они состоят из одинаковых слов.

Рассмотрим два предложения, которые предварительно прошли все этапы предобработки текста:

```
['побед', 'задума', 'введен', 'плат', 'пластиков', 'стаканчик', 'борт']
```

Рис. 6: Предложение 1

```
['введен', 'плат', 'углеродн', 'выброс']
```

Рис. 7: Предложение 2

В задачах NLP, каждое текстовое предложение называется документом, а несколько таких документов называют корпусом текстов или коллекцией. «Bag of words» создает словарь уникальных слов в корпусе.

Составим таблицу для нашего примера, где столбцы соответствуют уникальным словам, а строки документам. Запишем 1, если слово встречается в предложении, и 0, если нет.

| | побед | задума | введен | плат | пластико | стаканчик | борт | углеродн | выброс |
|---|---|---|---|---|---|---|---|---|---|
| пред. 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| пред. 2 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |

Далее составляется матрица $d \times n$, где $d$ — количество уникальных слов, а $n$ — количество документов. Для нашего примера матрица будет иметь размер $9 \times 2$. Для каждой ячейки в матрице высчитывается TF-IDF.

## 4.1. Term Frequency(TF)

Это метрика [8], которая показывает на сколько часто слово встречается в документе. Таким образом, оценивается важность слова в отдельном документе.

$$\mathrm{tf}(t, d) = \frac{n_t}{\sum_k n_k}$$

$n_t$ — число вхождений слова $t$ в документ, а в знаменателе общее число слов в документе.

В нашем примере для слова «плат» TF в первом документе будет равно $\frac{1}{7}$, а во втором $\frac{1}{4}$.

## 4.2. Inverse Document Frequency(IDF)

IDF [8] — это обратная частотность документов. Она измеряет важность слова относительно всех документов. Другими словами, если слово встречается часто во всех документах, оно не такое уж и важное.

$$\mathrm{idf}(t, D) = \log \frac{|D|}{|\{\, d_i \in D \mid t \in d_i \,\}|}$$

$|D|$ — число документов в коллекции;

$|\{\, d_i \in D \mid t \in d_i \,\}|$ — число документов из коллекции $D$, в которых встречается $t$ (когда $n_t \neq 0$).

## 4.3. Term Frequency — Inverse Document Frequency(TF-IDF)

TF-IDF [8] — метрика, которая используется для оценки важности слова в контексте документа, являющегося частью коллекции. Вес каждого слова прямо пропорционален частоте употребления этого слова в документе и обратно пропорционален частоте употребления слова во всех документах коллекции.

$$tf - idf(t, d, D) = tf(t, d) \times idf(t, D)$$

Для нашего примера матрица выглядела бы следующим образом:

| | побед | задума | введен | плат | пластиков | стаканчик | борт | углеродн | выброс |
|---|---|---|---|---|---|---|---|---|---|
| **пред. 1** | 0.4 | 0.4 | 0.29 | 0.29 | 0.4 | 0.4 | 0.4 | 0 | 0 |
| **пред. 2** | 0 | 0 | 0.41 | 0.41 | 0 | 0 | 0 | 0.58 | 0.58 |

# 5. Классификаторы

Следующим шагом после векторизации текста является его классификация. Для нашей задачи будем использовать три классификатора:

- Стохастический градиентный спуск (SGD);
- K-ближайших соседей (KNN);
- Случайный лес (Random Forest).

## 5.1. Стохастический градиентный спуск (SGD)

SGD — алгоритм, которые похож на градиентный спуск, но более оптимизированный. При стандартном градиентном спуске для корректировки параметров модели используется градиент. Для того чтобы менять параметры алгоритму требуется один раз целиком пройти по обучающимся данным. При стохастическом градиентном спуске значение градиента аппроксимируются градиентом функции стоимости, вычисленном только на одном элементе обучения. Затем параметры меняются прямо пропорционально приближенному градиенту. Таким образом параметры модели изменяются после каждого объекта обучения. Подробнее можно почитатать тут: [13].

К плюсам SGD можно причислить:

- Эффективность;
- Простота реализации.

К минусам:

- Требовательность (SGD необходим ряд гиперпараметров, таких как параметр регуляризации и количество итераций);
- Чувствительность к масштабированию признаков.

## 5.2. K-ближайших соседей (KNN)

Алгоритм K-ближайших соседей (KNN) [11] использует сходство признаков для прогнозирования новых значений. Другими словами, для каждой новой точки высчитывается расстояние до тренировочных точек. Выбирается k наиболее близких, и в качестве класса новой точки назначается самый популярный класс из k ближайших.

## 5.3. Случайный лес (Random Forest)

Случайный лес [6] — это классификатор, который использует ансамбль деревьев решений [10]. Каждое отдельное дерево решений генерируется с использованием следующих метрик: критерий прироста информации, отношение прироста и индекс Джини [12] для каждого признака. Любое такое дерево создается на основе независимой случайной выборки. Каждое дерево голосует за тот или иной класс, и в качестве окончательного результата выбирается самый популярный.

# 6. Реализация

## 6.1. Данные

Для задачи классификации текста на русском языке были взяты публичные данные [4] с новостного сайта Lenta.ru на 2 ГБ. Данные содержат новости разделеннные на 23 категории: «Россия», «Мир», «Экономика», «Спорт», «Культура», «Бывший СССР», «Наука и техника», «Интернет и СМИ», «Из жизни», «Дом», «Силовые структуры», «Ценности», «Бизнес», «Путешествия», «69-я параллель», «Крым», «Оружие», «ЧМ-2014», «МедНовости», «Сочи», «Библиотека», «Легпром», «Культпросвет».

Датасет предварительно сбалансирован. Данные с метками «Оружие», «ЧМ-2014», «МедНовости», «Сочи», «Библиотека», «Легпром», «Культпросвет», «Крым», «69-я параллель» были удалены из-за малого количества. Датасет разделен на тренировочные и тестовые данные.

## 6.2. Выбор ключевой метрики

После того как данные прошли чистку и предобработку, я начал тестирование моделей. В качестве ключевой метрики я выбрал f1-score [9], так как нам важен precision и recall [5]. Классов довольно много, поэтому для упрощения оценки метрики я выбрал macro avg. Он показывает средний f1-score по всем классам.

## 6.3. Тестирование моделей

Сначала я протестировал алгоритм SGD, KNN и Random Forest со стандартными параметрами. Результаты получились очень хорошие. С незначительным перевесом SGD показала себя лучше остальных.

## 6.4. Оптимизация модели

Следующим шагом я хотел оптимизировать модель, которая показала лучший результат на параметрах по умолчанию. С помощью

| | precision | recall | f1-score |
|---|---|---|---|
| Бизнес | 0.72 | 0.78 | 0.75 |
| Бывший СССР | 0.92 | 0.85 | 0.89 |
| Дом | 0.93 | 0.89 | 0.91 |
| Из жизни | 0.77 | 0.82 | 0.79 |
| Интернет и СМИ | 0.79 | 0.82 | 0.81 |
| Культура | 0.94 | 0.85 | 0.89 |
| Мир | 0.84 | 0.80 | 0.82 |
| Наука и техника | 0.90 | 0.89 | 0.89 |
| Путешествия | 0.86 | 0.82 | 0.84 |
| Россия | 0.65 | 0.84 | 0.74 |
| Силовые структуры | 0.86 | 0.78 | 0.82 |
| Спорт | 0.98 | 0.92 | 0.95 |
| Ценности | 0.87 | 0.89 | 0.88 |
| Экономика | 0.79 | 0.85 | 0.82 |
| accuracy | | | 0.84 |
| macro avg | 0.84 | 0.84 | 0.84 |
| weighted avg | 0.85 | 0.84 | 0.85 |

| | precision | recall | f1-score |
|---|---|---|---|
| Бизнес | 0.74 | 0.65 | 0.69 |
| Бывший СССР | 0.89 | 0.79 | 0.84 |
| Дом | 0.87 | 0.81 | 0.84 |
| Из жизни | 0.65 | 0.68 | 0.66 |
| Интернет и СМИ | 0.63 | 0.77 | 0.69 |
| Культура | 0.82 | 0.81 | 0.81 |
| Мир | 0.79 | 0.73 | 0.76 |
| Наука и техника | 0.83 | 0.84 | 0.83 |
| Путешествия | 0.74 | 0.79 | 0.77 |
| Россия | 0.63 | 0.71 | 0.66 |
| Силовые структуры | 0.76 | 0.76 | 0.76 |
| Спорт | 0.95 | 0.92 | 0.93 |
| Ценности | 0.80 | 0.88 | 0.84 |
| Экономика | 0.79 | 0.80 | 0.80 |
| accuracy | | | 0.78 |
| macro avg | 0.78 | 0.78 | 0.78 |
| weighted avg | 0.78 | 0.78 | 0.78 |

Рис. 8: Слева таблица для SGD, справа для KNN

| | precision | recall | f1-score |
|---|---|---|---|
| Бизнес | 0.66 | 0.77 | 0.71 |
| Бывший СССР | 0.87 | 0.81 | 0.84 |
| Дом | 0.89 | 0.87 | 0.88 |
| Из жизни | 0.76 | 0.70 | 0.73 |
| Интернет и СМИ | 0.69 | 0.77 | 0.72 |
| Культура | 0.87 | 0.83 | 0.85 |
| Мир | 0.78 | 0.75 | 0.76 |
| Наука и техника | 0.86 | 0.88 | 0.87 |
| Путешествия | 0.73 | 0.83 | 0.78 |
| Россия | 0.59 | 0.78 | 0.67 |
| Силовые структуры | 0.83 | 0.72 | 0.77 |
| Спорт | 0.98 | 0.90 | 0.93 |
| Ценности | 0.82 | 0.93 | 0.87 |
| Экономика | 0.82 | 0.78 | 0.80 |
| accuracy | | | 0.80 |
| macro avg | 0.80 | 0.81 | 0.80 |
| weighted avg | 0.81 | 0.80 | 0.80 |

Рис. 9: Таблица для Random Forest

RandomizedSearchCV [7] я начал перебор параметров для SGD и векторизатора.

RandomizedSearchCV — алгоритм, который перебирает указанные параметры и осуществляет перекрестную проверку. В качестве параметров были выбраны следующие: «loss», «class_weight», «penalty», «strip_accents» и «ngram_range».

- «loss» — функция потерь;
- «class_weight» — параметр, который отвечает за веса классов;

- «penalty» — регуляризатор. Для того чтобы модель не переобучалась;
- «strip_accents» — параметр векторизатора, который удаляет акценты и выполняет нормализацию символов на этапе предварительной обработки;
- «ngram_range» — параметр векторизатора, который указывает нижнюю и верхнюю границу значения n для n-грамм.

RandomizedSearchCV показал, что наилучшие результаты достигаются при следующих параметрах:

- «loss» = *perceptron*;
- «class_weight» = *None*;
- «penalty» = *l*2;
- «strip_accents» = *unicode*;
- «ngram_range» = $(1, 2)$.

Вот сами результаты:

| | precision | recall | f1-score |
|---|---|---|---|
| Бизнес | 0.80 | 0.82 | 0.81 |
| Бывший СССР | 0.93 | 0.90 | 0.91 |
| Дом | 0.94 | 0.93 | 0.94 |
| Из жизни | 0.82 | 0.85 | 0.84 |
| Интернет и СМИ | 0.81 | 0.87 | 0.84 |
| Культура | 0.90 | 0.91 | 0.91 |
| Мир | 0.87 | 0.84 | 0.86 |
| Наука и техника | 0.92 | 0.92 | 0.92 |
| Путешествия | 0.89 | 0.85 | 0.87 |
| Россия | 0.82 | 0.80 | 0.81 |
| Силовые структуры | 0.88 | 0.84 | 0.86 |
| Спорт | 0.97 | 0.95 | 0.96 |
| Ценности | 0.93 | 0.91 | 0.92 |
| Экономика | 0.84 | 0.89 | 0.87 |
| accuracy | | | 0.88 |
| macro avg | 0.88 | 0.88 | 0.88 |
| weighted avg | 0.88 | 0.88 | 0.88 |

Значение ключевой метрики выросло до 0.88, что является хорошим результатом.

# 7. Заключение

В соответствии с поставлеными задачи были изучены и успешно применены алгоритм предобработки текста и модели классификации текста.

Значение ключевой метрики у Стохастического градиентного спуска (SGD), K-ближайших соседей (KNN) и Случайного леса (Random Forest) равно 0.84, 0.78 и 0.81 соответсвенно. SGD была оптитмизирована, и значение ключевой метрики выросло до 0.88.

Код к данной задаче можно посмотреть на github [3].

# Список литературы


[1] Bird Klein и Loper. Natural Language Processing with Python. — 2019. — Access mode: http://www.datascienceassn.org/sites/default/files/Natural%20Language%20Processing%20with%20Python.pdf (online; accessed: 30.03.2022).

[2] Feature extraction // scikit-learn. — Access mode: https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction (online; accessed: 30.03.2022).

[3] ML в задачах классификации текста // github. — Access mode: https://github.com/valdik04/MachineLearning (online; accessed: 30.03.2022).

[4] News dataset from Lenta.Ru // kaggle. — Access mode: https://www.kaggle.com/datasets/yutkin/corpus-of-russian-news-articles-from-lenta (online; accessed: 30.03.2022).

[5] Precision-Recall // scikit-learn. — Access mode: https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html (online; accessed: 30.03.2022).

[6] Random Forest: прогулки по зимнему лесу // habr. — Access mode: https://habr.com/ru/post/320726/ (online; accessed: 30.03.2022).

[7] RandomizedSearchCV // scikit-learn. — Access mode: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html (online; accessed: 30.03.2022).

[8] Term frequency–inverse document frequency. — Access mode: https://wiki.loginom.ru/articles/tf-idf.html (online; accessed: 30.03.2022).

[9] f1_score // scikit-learn. — Access mode: https://scikit-learn.

org/stable/modules/generated/sklearn.metrics.f1_score.html (online; accessed: 30.03.2022).

[10] Дерево решений // habr. — Access mode: https://habr.com/ru/company/productstar/blog/523044/ (online; accessed: 30.03.2022).

[11] Классификатор KNN // habr. — Access mode: https://habr.com/ru/post/149693/ (online; accessed: 30.03.2022).

[12] Коэффициент Джини. Из экономики в машинное обучение // habr. — Access mode: https://habr.com/ru/company/ods/blog/350440/ (online; accessed: 30.03.2022).

[13] Обзор градиентных методов в задачах математической оптимизации // habr. — Access mode: https://habr.com/ru/post/413853/ (online; accessed: 30.03.2022).