# Основные сведения о машинном обучении

Машинное обучение – это класс компьютерных алгоритмов, характерной

чертой которых является не прямое решение задачи, а обучение за счёт

применения решений множества сходных задач [10].

В классическом программировании, люди создают правила (программу) и

данные, которые должны быть обработаны в соответствии с этими правилами, а

затем получают ответы. При машинном обучении человек вводит данные, а

также ответы, соответствующие этим данным, и на выходе получает правила,

которые затем могут быть применены к новым данным для получения

соответствующих ответов (рисунок 6).

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

Machine learning (ML) is a field devoted to understanding and building methods that let machines "learn" – that is, methods that leverage data to improve computer performance on some set of tasks.[1]

Machine learning algorithms build a model based on sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to do so.[2] Machine learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, agriculture, and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.[3][4]

Some implementations of machine learning use data and neural networks in a way that mimics the working of a biological brain.[8][9]

In its application across business problems, machine learning is also referred to as predictive analytics.

Learning algorithms work on the basis that strategies, algorithms, and inferences that worked well in the past are likely to continue working well in the future. These inferences can sometimes be obvious, such as "since the sun rose every morning for the last 10,000 days, it will probably rise tomorrow morning as well". Other times, they can be more nuanced, such as "X% of families have geographically separate species with color variants, so there is a Y% chance that undiscovered black swans exist".[10]

Machine learning programs can perform tasks without being explicitly programmed to do so. It involves computers learning from data provided so that they carry out certain tasks. For simple tasks assigned to computers, it is possible to program algorithms telling the machine how to execute all steps required to solve the problem at hand; on the computer's part, no learning is needed. For more advanced tasks, it can be challenging for a human to manually create the needed algorithms. In practice, it can turn out to be more effective to help the machine develop its own algorithm, rather than having human programmers specify every needed step.[11]

The discipline of machine learning employs various approaches to teach computers to accomplish tasks where no fully satisfactory algorithm is available. In cases where vast numbers of potential answers exist, one approach is to label some of the correct answers as valid. This can then be used as training data for the computer to improve the algorithm(s) it uses to determine correct answers. For example, to train a system for the task of digital character recognition, the MNIST dataset of handwritten digits has often been used.

Система машинного обучения обучается, а не программируется. Ей

предъявляется множество примеров, относящихся к задаче, и она находит

статистическую структуру в этих примерах, что в конечном итоге позволяет

системе выработать правила для автоматизации задачи [10].

Процесс машинного обучения может быть разбит на следующие этапы

подготовки и создания модели:

− подготовка данных (устранение дублирования, предобработка);

− выбор модели и ее обучение;

− оценка качества модели;

− использование модели для классификации новых примеров.

Рассмотрим более подробно такой подраздел машинного обучения, как

искусственные нейронные сети.

Machine learning is a subfield of artificial intelligence, which is broadly defined as the capability of a machine to imitate intelligent human behavior. Artificial intelligence systems are used to perform complex tasks in a way that is similar to how humans solve problems.

The goal of AI is to create computer models that exhibit “intelligent behaviors” like humans, according to Boris Katz, a principal research scientist and head of the InfoLab Group at CSAIL. This means machines that can recognize a visual scene, understand a text written in natural language, or perform an action in the physical world.

# Введение в искуственные нейронные сети

Искусственные нейронные сети – это математические модели, созданные

по подобию биологических нейронных сетей. Они являются одним из методов

машинного обучения [15].

Нейронная сеть основана на коллекции соединенных узлов, называемых

искусственными нейронами. Каждое соединение, подобно синапсам в

биологическом мозге, может передавать сигнал другим нейронам. Нейрон,

получающий сигнал, обрабатывает его и может отправить другой сигнал

соединенным с ним нейронам. Под сигналом понимается вещественное число.

Значение на выходе каждого нейрона подсчитывается путем применения

нелинейной функции к сумме его входов. Нелинейная функция используется для

обеспечения более сложных взаимодействий. Соединения между нейронами

называются гранями. У нейронов и граней есть веса, которые изменяются в

процессе обучения. Веса увеличивают или уменьшают силу сигнала. Нейроны

собраны в слои. Разные слои могут выполнять различные преобразования к

значениям, поступающим на вход. Сигналы перемещаются от первого

(входного) слоя к последнему (выходному) слою. Ниже на рисунке 7 приведено

схематическое представление полносвязной сети – сети, в которой каждый

нейрон предыдущего слоя связан с каждым нейронном следующего.

Нейронные сети обучаются путем обработки примеров, состоящих из

входного значения и результата. Обучение нейронной сети обычно производится

определением разницы между значением на выходе нейронной сети

(предсказанием) и результатом из примера. Такая разница называется значением

потерь. После этого сеть изменяет свои веса таким образом, чтобы уменьшить

это значение. С каждой новой итерацией значение потерь становится все меньше

и меньше, и, достигнув определенного критерия (например, точность превысила

определенный порог), обучение нейронной сети приостанавливается [10].

При обучении нейронной сети выполняется прямой и обратный ход.

Прямой ход.

1. На входной слой нейронной сети подается тензор и распространяется

по всей сети от слоя к слою.

2. Вычисляется выход сети.

Обратный ход.

1. Вычисляется разность между желаемым выходом сети и фактическим.

В результате получается значение потерь.

2. Полученный сигнал распространяется в обратном направлении

соединений с нейронами, и впоследствии корректируются веса сети с

целью минимизации ошибки.

Так как нейронной сети нужно очень большое количество данных для

обучения, уместить которое все сразу бывает чаще всего невозможно в ОЗУ

компьютера или видеопамяти видеокарты, то используют обучение при помощи

батчей. Батч – это малая часть набора данных, которая подается на вход

нейронной сети.

Обучение с помощью батчей происходит следующим образом.

1. Весь набор данных делится на определенное количество батчей.

2. На вход нейронной сети поступает батч, происходит прямой и

обратный ход.

3. На вход нейронной сети поступает следующий батч, происходит

прямой и обратный ход и т.д., пока на вход не поступят все батчи.

Когда на вход нейронной сети поступили все батчи и для каждого из них

произошли прямой ход и обратный ход, то говорят, что прошла 1 эпоха.

В основе обучения нейронных сетей лежит применение стохастического

градиентного спуска.

Если функция дифференцируема, то теоретически возможно найти ее

минимум аналитически: известно, что минимум функции – это точка, где

производная равна 0, поэтому достаточно найти все точки, где производная

обращается в 0, и проверить, в какой из этих точек функция имеет наименьшее

значение. Применительно к нейронной сети это означает аналитическое

нахождение комбинации значений весов, которая дает наименьшее значение для

функции потерь [10]. Этого можно достичь, решив уравнение (2) для 𝑊𝑊.

∇(𝑓𝑓)(𝑊𝑊) = 0, (2)

где ∇(𝑓𝑓)(𝑊𝑊) – градиент функции 𝑓𝑓, 𝑓𝑓 – функция нейронной сети, 𝑊𝑊 –

весовые коэффициенты нейронной сети.

Это полиномиальное уравнение из 𝑁𝑁 переменных, где 𝑁𝑁 – количество

параметров в сети. Хотя такое уравнение можно решить для 𝑁𝑁 = 2 или 𝑁𝑁 = 3,

это трудновыполнимо для реальных нейронных сетей, где число параметров

никогда не бывает меньше нескольких тысяч и часто может составлять

несколько десятков миллионов [16]. Вместо этого можно использовать

следующий алгоритм, называемый стохастическим градиентным спуском:

1. Взять батч данных, состоящих из примеров для обучения 𝑥𝑥 и

соответствующих меток 𝑦𝑦.

2. Выполнить прямой ход на взятом батче и получить предсказания 𝑦𝑦𝑝𝑝𝑝𝑝𝑝𝑝𝑝𝑝 .

3. Вычислить значение потерь сети на данном батче между 𝑦𝑦𝑝𝑝𝑝𝑝𝑝𝑝𝑝𝑝 и 𝑦𝑦.

4. Вычислить градиент значения потерь относительно параметров сети

(обратный ход).

5. Немного изменить параметры в направлении, противоположному

направлению градиента.

Процесс классификации с помощью нейронной сети делится на 2 этапа:

обучение и использование. Вначале на вход нейронной сети подаются примеры

для обучения. Далее сеть обучается, изменяя и настраивая весовые

коэффициенты, и уже с готовыми преобразованными весами используется как

классификатор.

Основными "строительными блоками" нейронных сетей являются

следующие сущности:

− слой – объединенные в одну группу нейроны;

− функция активации – функция, определяющая выходной сигнал

нейрона, основываясь на входном сигнале или наборе входных

сигналов. например, softmax, sigmoid, relu;

− функция потерь (целевая функция) – функция, значение которой

необходимо минимизировать при обучении нейронной сети;

− оптимизатор – алгоритм, определяющий то, как нейронная сеть будет

обновлять свои веса, основываясь на функции потерь.

Число нейронов, количество слоев, используемые функции активации и

оптимизатор называются гиперпараметрами сети.

При обучении стараются подобрать гиперпараметры так, чтобы достичь

максимальной производительности (высокой точности, минимального значения

потерь и т.д.).

Одним из ключевых понятий при работе с нейронными сетями является

понятие тензоров. Тензоры – это обобщение матриц на произвольное число

измерений. Матрица – это двумерный тензор.

Основные типы тензоров в нейронных сетях:

− скаляры (0D тензоры). Тензор, содержащий только одно число,

называется скаляром;

29

− векторы (1D тензоры). Массив чисел называется вектором, или

одномерным тензором. Считается, что одномерный тензор имеет ровно

одну ось;

− матрицы (двумерные тензоры). Массив векторов называется матрицей,

или двумерным тензором. Матрица имеет две оси (часто называемые

строками и столбцами);

− 3D тензоры. Если поместить матрицы в новый массив, то получится 3D

тензор, который можно визуально интерпретировать как куб чисел.

Помещая трехмерные тензоры в массив, можно создать

четырехмерный тензор, и так далее. Ниже на рисунке 8 приведена

визуальная интерпретация 3D тензора.

Тензор определяется тремя ключевыми атрибутами:

− количество осей (ранг). Например, трехмерный тензор имеет три оси, а

матрица – две оси;

− форма – это кортеж целых чисел, который описывает, сколько

измерений имеет тензор по каждой оси. Например, матрица с 3

строками и 5 столбцами имеет форму (3,5);

− тип данных. Тип данных, содержащихся в тензоре; например, тип

тензора может быть float32, uint8, float64 и так далее. В редких случаях

можно встретить тензор char.

Все операции, производимые нейронной сетью, она производит над

тензорами

Рассмотрев общий принцип работы искусственных нейронных сетей,

перейдем к их разновидности, специально созданной для распознавания

изображений.

# Предобработка текстового набора данных

В настоящее время для обработки текстов на естественном языке всё чаще применяются нейронные сети. Однако, чтобы получить более качественные результаты, текст, являющийся для них входными данными необходимо предобработать.

Текстовые данные для нейронной сети поступают в виде двумерного тензора (матрицы).

Различные методы предобработки можно поделить на следующие группы:

- Простые методы предобработки

- Методы нормализации

- Методы векторизации

- Методы балансирования текстового набора данных.

Рассмотрим каждую группу методов подробнее.

## Простые методы предобработки

К простым методам предобработки относятся следующие методы:

- Удаление дубликатов

- Удаление стоп-слов

- Удаление символов пунктуации

- Удаление чисел

- Удаление ведущих и конечных пробелов

- Приведение слов к единому регистру

Рассмотрим каждый из этих методов подробнее.

\*\*Нужно все писать применительно к русскому языку\*\*

\*\*Попробовать переформулировать текст с помощью бектранслейтинга\*\*

\*\*Методы помогают сети лучше найти связи и быстрее сойтись.\*\*

\*\*Удаление дубликатов\*\*

Набор данных может содержать в себе задублированные примеры. Удаление дубликатов направлено на нахождение таких примеров и их последующее удаление.

Например, ...

\*\*Удаление стоп-слов\*\*

Стоп-слова – это слова, которые часто встречаются в естественном языке, но при этом не несут большой смысловой нагрузки. Удаление стоп-слов направлено на нахождение таких слов в тексте и их последующее удаление.

К стоп-словам можно отнести предлоги, суффиксы, причастия, междометия, частицы и т.п.

There is no single universal list of stop words used by all natural language processing tools, nor any agreed upon rules for identifying stop words, and indeed not all tools even use such a list. Therefore, any group of words can be chosen as the stop words for a given purpose. The "general trend in [information retrieval] systems over time has been from standard use of quite large stop lists (200–300 terms) to very small stop lists (7–12 terms) to no stop list whatsoever".

Например, в русском языке это такие слова, как «а», «но», «и» и т.д. \*\*При удалении стоп–слов предложение «Термин interpunctio – римского происхождения, но само начало её неясно.» примет вид «Термин interpunctio – римского происхождения, само начало её неясно.».\*\*

\*\*Удаление символов пунктуации\*\*

Символ пунктуации - это ...

Чаще всего смысл текста остается понятным и после удаления из него символов пунктуации. Удаление символов пунктуации направлено на нахождение таких сиимволов в тексте и их последующее удаление.

\*\*Например, предложение «Термин interpunctio – римского происхождения, но само начало её неясно.» примет вид «Термин interpunctio римского происхождения но само начало её неясно».\*\*

\*\*Удаление чисел\*\*

Иногда числа в тексте не несут в себе особо значимой информации, поэтому их можно удалить. Удаление чисел направлено на нахождение чисел в тексте и их последующее удаление.

Например, ...

\*\*Удаление излишних пробельных символов\*\*

Пробельные символы - это ...

Как правило, слова в тексте разделяются с помощью единственного пробела. Кроме того, текст может содержать в себе другие пробельные символы, которые по сути не несут в себе особо значимой информации и могут быть заменены на одиночный пробел.

Ведущие и конечные пробелы во фрагменте так же в большинстве случае не играют роли, поэтому их можно удалить.

Удаление излишних пробельных символов направлено на:

- нахождение последовательности из двух и более пробелов и их замену на одиночный пробел

- замену пробельных символов, отличных от пробела на пробел

- удаление ведущих и конечных пробелов

Например, ...

\*\*Приведение слов к единому регистру\*\*

Слова в тексте могут находиться в различном регистре. \*\*Как именно в русском языке?\*\* Например, если они стоят в начале предложения, это имя собственное и т.д.

Как правило, в русском языке, можно изменить регистр, в котором написано слово и при этом оно не потеряет своего значения. При составлении словаря слова считались бы разными, а это поможет их считать одинаковыми.

При токенизации составлении словаря слова с разным регистром являются для токенизатора разными, хотя они по сути одно и то же, чтобы этого избежать, слова приводят к одному регистру

Например, ...

Может возникнуть вопрос, к какому регистру лучше приводить слова? К верхнему или нижнему? В основном не играет значения, но по-скольку в естественном языке большинство букв пишется в предложении пишется в нижнем регистре, то с точки зрения вычислительных затрат эффективнее приводить слова к нижнему регистру.

https://www.petefreitag.com/item/175.cfm

## Методы нормализации

\*\*Что такое нормализация текста https://en.wikipedia.org/wiki/Text\_normalization https://towardsdatascience.com/text-normalization-for-natural-language-processing-nlp-70a314bfa646\*\*

\*\*Как производится токенизация\*\*

Прежде чем применять к тексту методы нормализации, его необходимо токенизировать.

Токенизация - это ...

Токен - это ...

Например, ...

### Стемминг

\*\*Что такое стемминг?\*\*

\*\*Как производится?\*\*

\*\*Пример\*\*

\*\*Плюсы и минусы\*\*

\*\*Какой в NLTK алгоритм стемминга?\*\*

In linguistic morphology and information retrieval, stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form—generally a written word form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root.

Существует несколько типов алгоритмов стемминга, которые различаются по отношению производительности, точности, а также как преодолеваются определённые проблемы стемминга.

Алгоритмы поиска

Простой стеммер ищет флективную форму в таблице поиска. Преимущества этого подхода заключается в его простоте, скорости, а также лёгкости обработки исключений. К недостаткам можно отнести то, что все флективные формы должны быть явно перечислены в таблице: новые или незнакомые слова не будут обрабатываться, даже если они являются правильными (например, iPads ~ iPad), а также проблемой является то, что таблица поиска может быть очень большой. Для языков с простой морфологией наподобие английского размеры таблиц небольшие, но для сильно флективных языков (например, турецкого) таблица может иметь сотни возможных флективных форм для каждого корня.

Таблицы поиска, используемые в стеммерах, как правило, генерируются в полуавтоматическом режиме. Например, для английского слова «run» автоматически будут сгенерированы формы «running», «runs», «runned» и «runly». Последние две формы являются допустимыми конструкциями, но они вряд ли появятся в обычном тексте на английском языке.

Алгоритм поиска может использовать предварительную частеречную разметку, чтобы избежать такой разновидности ошибки лемматизации, когда разные слова относят к одной лемме (overstemming)[2].

Алгоритмы усечения окончаний

Алгоритмы усечения окончаний не используют справочной таблицы, которая состоит из флективных форм и отношений корня и формы. Вместо этого, как правило, хранится меньший список «правил», который используется алгоритмами, учитывая форму слова, чтобы найти его основу[3]. Некоторые примеры правил выглядят следующим образом:

если слово оканчивается на 'ed', удалить 'ed'

если слово оканчивается на 'ing', удалить 'ing'

если слово оканчивается на 'ly', удалить 'ly'

Алгоритмы усечения окончаний гораздо эффективнее, чем алгоритмы полного перебора. Для разработки таких алгоритмов нужен программист, который достаточно хорошо разбирается в лингвистике, в частности морфологии, а также умеет кодировать «правила усечения». Алгоритмы усечения окончаний неэффективны для исключительных ситуаций (например, 'ran' и 'run'). Решения, полученные алгоритмами усечения окончаний, ограничиваются теми частями речи, которые имеют хорошо известные окончания и суффиксы с некоторыми исключениями. Это является серьёзным ограничением, так как не все части речи имеют хорошо сформулированный набор правил. Лемматизация пытается снять это ограничение.

Алгоритмы усечения префикса также могут быть реализованы. Однако не во всех языках слова имеют приставки и суффиксы.

Дополнительные критерии алгоритмов

Алгоритмы усечения окончаний могут отличаться в результатах по целому ряду причин. Одной из таких причин является особенность алгоритма: должно ли быть слово на выходе алгоритма реальным словом в данном языке. Некоторые подходы не требуют наличия слова в соответствующей языковой лексике. Кроме того, некоторые алгоритмы ведут базу данных всех известных морфологических корней, которые существуют как реальные слова. Данные алгоритмы проверяют наличие терма в базе данных для принятия решения. Как правило, если терм не найден, выполняются альтернативные действия. Данные альтернативные действия могут использовать несколько другие критерии для принятия решения. Несуществующий терм может послужить применению альтернативных правил усечения.

Может быть так, что два или более правила усечения применяются для одного и того же входного терма, что создаёт неопределённость относительно того, какое правило применить. Алгоритм может определить приоритет выполнения таких правил (с помощью человека или стохастическим способом). Или алгоритм может отклонить одно из правил, если оно приводит к несуществующему терму, тогда как другое — нет. Например, для английского терма «friendlies» алгоритм может определить суффикс «ies», применить соответствующее правило и получить результат «friendl». Терм «friendl», скорее всего, не будет найден в лексиконе и, следовательно, данное правило будет отвергнуто.

Одним из улучшений алгоритмов усечения окончаний является использование подстановки суффиксов и окончаний. Подобно правилу усечения, правило подстановки заменяет суффикс или окончание альтернативным. Например, может существовать правило, которое заменяет «ies» на «y». Поскольку правила усечения приводят к несуществующему терму в лексиконе, то правила подстановки решают эту проблему. В этом примере, «friendlies» преобразуется в «friendly» вместо «friendl».

Обычно применение данных правил носит циклический или рекурсивный характер. После первого применения правила подстановки для этого примера алгоритм производит выбор следующего правила для терма «friendly», в результате которого будет выявлено и признано правило усечения суффикса «ly». Таким образом, терм «friendlies» с помощью правила подстановки становится термом «friendly», который после применения правила усечения, становится термом «friend».

Данный пример помогает продемонстрировать разницу между методом, основанным на правилах, и методом «грубой силы». С помощью полного перебора алгоритм будет искать терм «friendlies» в наборе из сотни тысяч флективных словоформ и в идеале найдёт соответствующую основу «friend». В методе, основанном на правилах, происходит последовательное выполнение правил, в результате чего получается то же решение. Скорее всего, метод на основе правил будет быстрее.

Аффикс-стеммеры

В лингвистике самыми распространёнными терминами для обозначения аффиксов являются суффикс и префикс. В дополнение к подходам, которые обрабатывают суффиксы или окончания, некоторые из них также обрабатывают префиксы. Например, для английского слова indefinitely данный метод определит, что конструкция «in», стоящая в начале слова, является префиксом и может быть удалена для получения основы слова. Многие из методов, упомянутых выше, также используют данный подход. Например, алгоритм усечения окончаний может обрабатывать как суффиксы и окончания, так и префиксы, тогда он будет носить другое название и следовать данному подходу. Исследования аффикс-стеммеров для нескольких европейских языков можно найти в публикации (Jongejan et al 2009).

Алгоритмы лемматизации

Более сложным подходом к решению проблемы определения основы слова является лемматизация. Чтобы понять, как работает лемматизация, нужно знать, как создаются различные формы слова. Большинство слов изменяется, когда они используются в различных грамматических формах. Конец слова заменяется на грамматическое окончание, и это приводит к новой форме исходного слова. Лемматизация выполняет обратное преобразование: заменяет грамматическое окончание суффиксом или окончанием начальной формы[4].

Также лемматизация включает определение части речи слова и применение различных правил нормализации для каждой части речи. Определение части речи происходит до попытки найти основу, поскольку для некоторых языков правила стемминга зависят от части речи данного слова.

Этот подход крайне зависим от точного определения лексической категории (часть речи). Хотя и существует частичное совпадение между правилами нормализации для некоторых лексических категорий, указание неверной категории или неспособность определить правильную категорию сводит на нет преимущество этого подхода над алгоритмом усечения окончаний. Основная идея заключается в том, что если стеммер имеет возможность получить больше информации об обрабатываемом слове, то он может применять более точные правила нормализации.

Подход Ripple-down rules

Первоначально Ripple-down rules был разработан для приобретения знаний и обслуживания систем, основанных на правилах. В данном подходе знания приобретаются на основе текущего контекста и постепенно добавляются. Правила создаются для классификации случаев, соответствующих определённому контексту.

В отличие от стандартных правил классификации, Ripple-down Rules использует исключения из существующих правил, поэтому изменения связаны только с контекстом правила и не влияют на другие. Инструменты приобретения знаний помогают найти и изменить конфликтующие правила. Приведём простой пример правила Ripple-down:

if a ^ b then c

except if d then e

else if f ^ g then h

Данное правило можно проинтерпретировать так: «если a и b истина, то мы принимаем решение c, за исключением случая, когда d не истина. Если d истина (исключение), то принимаем решение e. Если a и b не истина, то мы переходим к другому правилу и принимаем решение h, если f и g истина». Эта форма правил очень хорошо решает задачу лемматизации[5].

Чтобы создать исключение из правила, алгоритм должен сначала определить слово, которое индуцировало данное исключение. После этого определяются различия между двумя словами. Условие исключения правила будет соответствовать этим различиям.

Стохастические алгоритмы

Стохастические алгоритмы связаны с вероятностным определением корневой формы слова. Данные алгоритмы строят вероятностную модель и обучаются с помощью таблицы соответствия корневых и флективных форм. Эта модель обычно представлена в виде сложных лингвистических правил, аналогичных по своему характеру правилам, использующимся в алгоритмах усечения окончаний и лемматизации. Стемминг выполняется посредством ввода изменённых форм для обучения модели и генерацией корневой формы в соответствии с внутренним набором правил модели, за исключением того, что решения, связанные с применением наиболее соответствующего правила или последовательности правил, а также выбором основы слова, применяются на основании того, что результирующее верное слово будет иметь самую высокую вероятность (неверные слова имеют наименьшую вероятность).

Некоторые алгоритмы лемматизации имеют стохастический характер в том смысле, что слово может принадлежать нескольким частям речи с разной вероятностью. Эти алгоритмы также могут учитывать окружающие слова, называемые контекстом. Контекстно-свободные грамматики не учитывают какую-либо дополнительную информацию. В любом случае, после присвоения вероятности каждой возможной части речи, выбирается часть речи с большей вероятностью, а также соответствующие ей правила для получения нормализованной формы.

Статистические алгоритмы

Анализ N-грамм

Некоторые алгоритмы стемминга используют анализ N-грамм, для того чтобы выбрать подходящую основу для слова[6].

Стемминг на основе корпуса текстов

Одним из главных недостатков классических стеммеров (например, стеммера Портера) является то, что они часто не различают слова со схожим синтаксисом, но совершенно с разными значениями. Например, «news» и «new» в результате стемминга сведутся к основе «new», хотя данные слова принадлежат к разным лексическим категориям. Другая проблема заключается в том, что некоторые алгоритмы стемминга могут быть пригодны для одного корпуса и вызывать слишком много ошибок в другом. Например, слова «stock», «stocks», «stocking» и т. д. будут иметь особое значение в текстах газеты The Wall Street Journal. Основная идея стемминга на основе корпуса текстов состоит в создании классов эквивалентности для слов классических стеммеров, а затем «разбить» некоторые слова, объединённые на основе их встречаемости в корпусе. Это также помогает предотвратить хорошо известные конфликтные ситуации алгоритма Портера, например, как «policy/police», так как шанс встретить данные слова вместе является довольно низким[7].

Алгоритмы сопоставления

Такие алгоритмы используют базу данных основ (например, набор документов, содержащие основы слов). Данные основы не обязательно соответствуют обычным словам, в большинстве случаев основа представляет собой подстроку (например, для английского языка «brows» является подстрокой в словах «browse» и «browsing»). Для того, чтобы определить основу слова, алгоритм пытается сопоставить его с основами из базы данных, применяя различные ограничения, например, на длину искомой основы в слове относительно длины самого слова (так например, короткий префикс «be», который является основой таких слов, как «be», «been» и «being», не будет являться основой слова «beside»).

Гибридные подходы

Гибридные подходы используют два или более методов, описанных выше. Простым примером является алгоритм, использующий суффиксное дерево, который в начале своей работы использует таблицы поиска для получения первоначальных данных с помощью полного перебора. Тем не менее, вместо того, чтобы хранить весь комплекс отношений между словами для определённого языка, таблица поиска используется для хранения небольшого числа «частых исключений» (например, для английского языка «ran => run»). Если слово отсутствует в списке исключения, применяется алгоритмы усечения окончаний или лемматизации для получения результата.

Стемминг русского языка

Русский язык относится к группе флективных синтетических языков, то есть языков, в которых преобладает словообразование с использованием аффиксов, сочетающих сразу несколько грамматических значений (например, добрый — окончание ый указывает одновременно на единственное число, мужской род и именительный падеж), поэтому данный язык допускает использование алгоритмов стемминга. Русский язык имеет сложную морфологическую изменяемость слов, которая является источником ошибок при использовании стемминга. В качестве решения данной проблемы можно использовать наряду с классическими алгоритмами стемминга алгоритмы лемматизации, которые приводят слова к начальной базовой форме.

Рассмотрим наиболее популярные реализации стеммеров, основывающиеся на различных принципах и допускающие обработку несуществующих слов для русского языка.

Стеммер Портера

Основная идея стеммера Портера заключается в том, что существует ограниченное количество словообразующих суффиксов, и стемминг слова происходит без использования каких-либо баз основ: только множество существующих суффиксов и вручную заданные правила.

Алгоритм состоит из пяти шагов. На каждом шаге отсекается словообразующий суффикс и оставшаяся часть проверяется на соответствие правилам (например, для русских слов основа должна содержать не менее одной гласной). Если полученное слово удовлетворяет правилам, происходит переход на следующий шаг. Если нет — алгоритм выбирает другой суффикс для отсечения. На первом шаге отсекается максимальный формообразующий суффикс, на втором — буква «и», на третьем — словообразующий суффикс, на четвёртом — суффиксы превосходных форм, «ь» и одна из двух «н»[13].

Данный алгоритм часто обрезает слово больше необходимого, что затрудняет получение правильной основы слова, например кровать->крова (при этом реально неизменяемая часть — кроват, но стеммер выбирает для удаления наиболее длинную морфему). Также стеммер Портера не справляется со всевозможными изменениями корня слова (например, выпадающие и беглые гласные).

Stemka

Данный алгоритм стемминга (анализатор) разработан Андреем Коваленко в 2002 году. Он основан на вероятностной модели: слова из обучающего текста разбираются анализатором на пары «последние две буквы основы» + «суффикс» и если такая пара уже присутствует в модели — увеличивается её вес, иначе она добавляется в модель. После чего полученный массив данных ранжируется по убыванию веса и модели, вероятность реализации которых составляет менее 1/10000, отсекаются. Результат — набор потенциальных окончаний с условиями на предшествующие символы — инвертируется для удобства сканирования словоформ «справа налево» и представляется в виде таблицы переходов конечного автомата. При разборе слово сканируется по построенным таблицам перехода. К алгоритму также было добавлено специальное правило, гласящее, что неизменяемая основа должна содержать как минимум одну гласную[14].

Представленный анализатор доступен в исходных текстах и может использоваться в свободной форме с условием ссылки на источник[15][16].

MyStem

Стеммер MyStem разработан Ильёй Сегаловичем в 1998. Сейчас является собственностью компании Яндекс[17]. На первом шаге при помощи дерева суффиксов во входном слове определяются возможные границы между основой и суффиксом, после чего для каждой потенциальной основы (начиная с самой длинной) бинарным поиском по дереву основ проверяется её наличие в словаре либо нахождение наиболее близких к ней основ (мерой близости является длина общего «хвоста»). Если слово словарное — алгоритм заканчивает работу, иначе — переходит к следующему разбиению.

Если вариант основы не совпадает ни с одной из «ближайших» словарных основ, то это означает, что анализируемое слово с данным вариантом основы в словаре отсутствует. Тогда по имеющейся основе, суффиксу и модели «ближайшей» словарной основы генерируется гипотетическая модель изменения данного слова. Гипотеза запоминается, а если она уже была построена ранее — увеличивает свой вес. Если слово так и не было найдено в словаре — длина требуемого общего окончания основ уменьшается на единицу, идёт просмотр дерева на предмет новых гипотез. Когда длина общего «хвоста» достигает 2, поиск останавливается и идёт ранжирование гипотез по продуктивности: если вес гипотезы в пять и более раз меньше самого большого веса — такая гипотеза отсеивается. Результатом работы стеммера является получившийся набор гипотез для несуществующего или одна гипотеза для словарного слова[18].

Стеммер может быть использован для коммерческих целей; исключениями являются: создание и распространение спама, поисковая оптимизация сайтов и разработка продуктов и сервисов, аналогичных сервисам и продуктам компании Яндекс[17]. Исходные коды не распространяются[19]. Для установки достаточно скачать и распаковать архив[20].

### Лемматизация

\*\*Что такое?\*\*

\*\*Как производится?\*\*

\*\*Пример\*\*

\*\*Плюсы и минусы\*\*

\*\*Какой в PyMorphy алгоритм лемматизации?\*\*

Лемматизация – это приведения словоформы к лемме – её словарной форме. В русском языке словарными формами считаются следующие морфологические формы:

• для существительных – именительный падеж, единственное число;

• для прилагательных – именительный падеж, единственное число, мужской род;

• для глаголов, причастий, деепричастий – глагол в неопределённой форме несовершенного вида.

Например, при лемматизации слово «работы» примет вид «работа».

Lemmatisation (or lemmatization) in linguistics is the process of grouping together the inflected forms of a word so they can be analysed as a single item, identified by the word's lemma, or dictionary form.[1]

In computational linguistics, lemmatisation is the algorithmic process of determining the lemma of a word based on its intended meaning. Unlike stemming, lemmatisation depends on correctly identifying the intended part of speech and meaning of a word in a sentence, as well as within the larger context surrounding that sentence, such as neighboring sentences or even an entire document. As a result, developing efficient lemmatisation algorithms is an open area of research.[2][3][4]

Lemmatisation is closely related to stemming. The difference is that a stemmer operates on a single word without knowledge of the context, and therefore cannot discriminate between words which have different meanings depending on part of speech. However, stemmers are typically easier to implement and run faster.

Алгоритм лемматизации основан на поиске наиболее подходящего варианта слова по словарю. При анализе текстовой информации обычно используются данные, полученные после процесса токенизации, подразумевающего разделение текста на отдельные слова или предложения. После сопоставления со словарем все словоформы одного слова заменяются на одно конкретное значение.

В языках со сложным словообразованием (например, русском) может потребоваться помимо стандартных словарей использовать дополнительные, учитывающие специфику речи. Отдельно к процессу лемматизации подключаются словари сленга, аббревиатуры и сокращений.

## Методы векторизации

Нейронные сети оперируют числами, поэтому, прежде чем подать текст на вход, его необходимо векторизовать.

\*\*Что такое векторизация?\*\*

\*\*Подробнее, зачем нужна векторизация\*\*

К основным методам векторизации относятся:

- Мешок слов

- Индексы (? не помню, как правильно, надо посмотреть)

- TF-IDF

- Embedding

### Мешок слов

\*\*Что такое?\*\*

\*\*Как производится?\*\*

\*\*Пример\*\*

\*\*Плюсы и минусы\*\*

https://www.mygreatlearning.com/blog/bag-of-words/

https://www.deepset.ai/blog/what-is-text-vectorization-in-nlp

https://www.oreilly.com/library/view/applied-text-analysis/9781491963036/ch04.html

https://www.enjoyalgorithms.com/blog/word-vector-encoding-in-nlp

### Индексы

\*\*Что такое?\*\*

\*\*Как производится?\*\*

\*\*Пример\*\*

\*\*Плюсы и минусы\*\*

### TF-IDF

\*\*Что такое?\*\*

\*\*Как производится?\*\*

\*\*Пример\*\*

\*\*Плюсы и минусы\*\*

### Embedding

Тут два каких то метода

\*\*Что такое?\*\*

\*\*Как производится?\*\*

\*\*Пример\*\*

\*\*Плюсы и минусы\*\*

## Методы балансирования текстового набора данных

Однако при ее решении стоит учитывать проблему несбалансированности

классов.

Несбалансированность классов – это проблема, при которой на один

пример в одном классе приходятся десятки, сотни или даже тысячи примеров в

другом.

Использование наборов данных с несбалансированными классами

представляет собой проблему для моделирования, т.к. большинство алгоритмов

машинного обучения спроектированы вокруг предположения о равном

количестве экземпляров для каждого класса [22]. Это ведет к тому, ч то модель

имеет крайне слабую обобщающую способность, особенно для класса,

находящегося в меньшинстве.

Если использовать для обучения нейронной сети данные,

несбалансированные по классам, например, набор данных в котором 60000

изображений представляют из себя рентгенограмму без патологий и лишь 100

изображений содержат патологию, то нейронная сеть будет в большинстве

43

случаев предсказывать лишь 1-ый класс, в то время как 2-ой будет

предсказываться гораздо реже.

Для решения проблемы несбалансированности классов можно использовать следующие методы:

- Даунсемплинг

- Оверсемплинг

- Приведение количества примеров к среднему значению

Рассмотрим каждый из примеров подробнее.

\*\*Зачем нужно (взять из прошлого диплома)\*\*

### Даунсемплинг

\*\*Что такое?\*\*

\*\*Как производится?\*\*

\*\*Пример\*\*

\*\*Плюсы и минусы\*\*

### Оверсемплинг

\*\*Что такое?\*\*

\*\*Как производится?\*\*

\*\*Пример\*\*

\*\*Плюсы и минусы\*\*

### Приведение количества примеров к среднему значению

\*\*Что такое?\*\*

\*\*Как производится?\*\*

\*\*Пример\*\*

\*\*Плюсы и минусы\*\*

# Архитектуры нейронных сетей для классификации текста

Для классификации текста можно использовать различные архитектуры нейронных сетей, однако наиболее частоиспользуемыми \*\*найти ссылку на то, какие часто используются, допустим, на Kaggle\*\* являются RNN и LSTM. Рассмотрим их подробнее.

## RNN

\*\*Что такое?\*\*

\*\*Каков принцип работы?\*\*

\*\*Картинки\*\*

\*\*Плюсы и минусы\*\*

## LSTM

\*\*Что такое?\*\*

\*\*Каков принцип работы?\*\*

\*\*Картинки\*\*

\*\*Плюсы и минусы\*\*

# Решение задачи классификации

\*\*Постановка задачи классификации\*\*

\*\*Оценка качествая обучения модели для решения задачи классификации - Точность и еще какие-то метрики\*\*