УДК 338.27

Григорян Эмиль Грачяевич

бакалавр направления подготовки
«Информатика и вычислительная техника»
Национальный исследовательский ядерный университет «МИФИ»
Россия, Москва

Emil H. Grigoryan

Bachelor of studies "Informatics and computer engineering"
National Research Nuclear University MEPhI
Russia, Moscow

Паршин Михаил Николаевич

бакалавр направления подготовки
«Информатика и вычислительная техника»
Национальный исследовательский ядерный университет «МИФИ»
Россия, Москва
zojloverr@gmail.com

Mikhail N. Parshin

Bachelor of the field of study "Informatics and Computer Engineering"
National Research Nuclear University MEPhI
Russia, Moscow

МЕТОДЫ NLP ДЛЯ ПРЕДОБРАБОТКИ ТЕКСТОВЫХ ДАННЫХ И ВЫДЕЛЕНИЯ ПРИЗНАКОВ

NLP METHODS FOR TEXT DATA PROCESSING AND SIGNALING

Аннотация

Обработка естественного языка или NLP является одним из наиболее перспективных направлений развития искусственного интеллекта. С ростом производительности компьютеров появляется все большее количество задач, связанных с анализом текстовой информации, которые могут быть эффективно решены при помощи технологий искусственного интеллекта.

Abstract

Natural language processing or NLP is one of the most promising areas for the development of artificial intelligence. With the growth of computer performance, an increasing number of tasks related to the analysis of textual information appear, which can be effectively solved using artificial intelligence technologies.

Ключевые слова:

искусственный интеллект, математическая лингвистика, закон Ципфра

Keywords:

artificial intelligence, mathematical linguistics, Zipfra's law

Введение

Обработкой естественного языка называют направление искусственного интеллекта и математической лингвистики. Основными задачами NLP являются: синтез текстов и проблемы компьютерного анализа. Говоря иными словами: NLP дает машинам возможность понимать и извлекать значение из различных языков и находится на пересечении науки о данных и человеческого языка. Анализ текстовой информации находит применение в обширном круге задач. Например, создание чат-ботов и виртуальных ассистентов, анализ отзывов о товарах или комментариев в интернете.

NLP ищет закономерности в текстах, обучаясь на данных, которые находятся в огромном количестве в интернете. NLP позволяет применять стандартные модели

машинного обучения к текстовым данным и записям речи. Например, логистическая регрессия может использоваться для классификации текстов [1].

Целью данной работы является обзор основных методов, которые используются в NLP для предобработки текстовых данных и выделения значимых признаков. В качестве данных, на которых показано как работают те или иные методы, использовались отзывы о фильмах пользователей сервиса КиноПоиск.

Предобработка текста

Этап предобработки является очень важным, потому что в текстах могут часто встречаются ошибки, лишние символы и прочие аномалии. Для человека понять смысл текстовых данных не является проблемой, ведь с детства в нас были заложены определенные правила, используя которые мы способны анализировать написанный текст, однако для моделей машинного обучения данный формат не используется. Рассмотрим различные способов приведения текста в вид, который будет более информативным для моделей машинного обучения.

Токенизацией называется процесс разбиения текста на слова (токены, термы и т.д.). На данном этапе могут быть удалены лишние символы, разделены «слипшиеся» слова.

Рассмотрим некоторые способы такого разбиения для выбора наилучшего метода для данной задачи. Все способы токенизации будут продемонстрированы с помощью одной и той же строки, приведенной на рисунке 1.

```
string = """Вот эдакая какая-нибудь глупость, какая-нибудь пошлейшая мелочь,
весь замысел может испортить!... Мелочи, мелочи главное!.. Вот эти-то мелочи и
губят всегда и всё...:( Встреча была назначена на 19.00"""
```

Рисунок 1 – Строка для изучения способов токенизации

Мощным средством токенизации являются регулярные выражения [3], которые позволяют описать некоторые шаблоны и вычленить их из текста. Регулярные выражение поддерживаются многими языками программирования. На рисунке 2 показан пример регулярного, которое вычленяет из текста слова, написанные кириллицей и разделенные любыми знаками препинания.

```
1 print(re.findall("[ёЁа-яА-Я]+", string))
```

```
['Вот', 'эдакая', 'какая', 'нибудь', 'глупость', 'какая', 'нибудь', 'пошлейшая', 'мело
чь', 'весь', 'замысел', 'может', 'испортить', 'Мелочи', 'мелочи', 'главное', 'Вот', 'э
ти', 'то', 'мелочи', 'и', 'губят', 'всегда', 'и', 'всё', 'Встреча', 'была', 'назначена
', 'на']
```

Рисунок 2 – Регулярное выражение по символам русского алфавита

Другим подходом к решению данной задачи может оказаться применение токенизатора из библиотеки Nltk. Он был разработан для анализа социальных сетей, поэтому умеет работать с различными вспомогательными символами или устоявшимися выражениями [4]. Пример его работы показан на рисунке 3.

```
1 print(nltk.tokenize.casual_tokenize(string))
```

```
['Вот', 'эдакая', 'какая-нибудь', 'глупость', ',', 'какая-нибудь', 'пошлейшая', 'мелоч
ь', ',', 'весь', 'замысел', 'может', 'испортить', '!', '...', 'Мелочи', ',', 'мелочи',
'главное', '!', '..', 'Вот', 'эти-то', 'мелочи', 'и', 'губят', 'всегда', 'и', 'всё', '
...', ':(', 'Встреча', 'была', 'назначена', 'на', '19.00']
```

Рисунок 3 – Использование библиотеки Nltk

Из рисунка 3 мы видим, что довольно хорошо были разделены вспомогательные символы.

Пользу в качество модели также может принести нормализация текста по регистру. Для этого рассмотрим строку, показанную на рисунке 4.

```
1 string = "Терпение и время, вот мои воины-богатыри!» — думал Кутузов"
```

Рисунок 4 – Строка демонстрации важности приведения к нижнему регистру

В приведенном выше примере "Терпение" написано с заглавной буквы, потому что стоит в начале предложения, но при этом имеет меньшее значение, чем "Кутузов". Исходя из того, что слова, которые стоят в начале предложения чаще имеют не такую большую ценность для конкретного текста, чем имена собственные, можно привести все слова к нижнему регистру, что продемонстрировано на рисунке 5.

```
1 print(string.lower())
```

```
терпение и время, вот мои воины-богатыри!» — думал кутузов
```

Рисунок 5 – Приведение строки к нижнему регистру

Частота появления слов в коллекциях текстов

Анализ частотности текста, наряду с нормализацией по регистру, является одним из сравнительно простых методов обработки текста на естественном языке (NLP). Но, несмотря на это, бывает полезно учитывать данную особенность при решении реальных задач.

К наиболее часто встречающимся словам в русском языке можно отнести так называемые вспомогательные части речи: предлоги, суффиксы, причастия, междометия, цифры, частицы и т.д., они присутствуют почти во всех текстах, хотя и не несут никакой полезной информации для модели.

Анализ частотности слов в разных языках заставляет задуматься о том, какое количество самых часто встречающихся слов в том или ином языке можно отбросить при решении задач, связанных с машинным обучением. Частично помочь с ответом на этот вопрос может закон Ципфра [2], который показывает определенные закономерности частотности слов в языках. Суть закона заключается в том, что если расположить слова в порядке убывания их частотности, пронумеровать и отобразить на графике, то на нем можно будет увидеть распределение Парето [4].

Рисунок 6 – Частотные зависимости

На рисунке 7 можно наблюдать зависимость частоты слова от их ранга, где рангом будем называть порядковый номер слова в таблице частотности. Поскольку на рисунке 6 не показано какие конкретно слова стоят в том или ином месте графика, на рисунке 7 покажем частоту использования некоторых слов в русском языке.

word	count
и	99066
в	88906
что	60461
я	49177
это	48070
на	41620
не	41373

Рисунок 7 – Частотность слов

Действительно, вспомогательные части речи являются наиболее часто встречающимися.

Подходы к формированию признаков

После проведения токенизации и нормализации текстовые данные необходимо перевести в виде векторов признаков. Для этого существуют различные способы, в данной работе будет рассмотрены следующие из них: Мешок слов, Tf-idf и Word2vec.

Мешок слов

Данный метод носит название «мешок слов», потому что в нем не используется никакая информация о порядке слов или их структуре. Для создания признаков данная модель использует только информацию о нахождении того или иного слова в тексте, но упускает довольно важную информацию – положение слова в нем [1]. «Мешок слов» строится на одном простом соображении, что, если документы имеют похожее содержание, то они являются похожими. Минусами данного подхода являются то, что мы никак не учитываем грамматические особенности текста и полностью теряем контекст слов, который, как будет показано дальше, является очень важным, также нет возможности восстановить порядок слов. Стоит заменить, что еще одним важным упущением метода является то, что мы потеряли грамматику, хотя во многих языках, таких, как английский используется строгий порядок слов. Стоит также понимать, что количество вхождения определенного слова в текст в сильной взаимосвязи с объемом текста, метод, который описывается следующим решает данную проблему.

Векторизация на основе tf-idf

TD-IDF - статистическая мера, которая используется для оценки важности слова в контексте документа, являющегося частью набора документов [1].

Для того, чтобы решить проблему зависимости важности слов от объема текста, которая была в предыдущем методе, вводится понятие TF (term frequency - частота слова).

$$\mathbf{tf}(\mathbf{w}, \mathbf{d}) = \frac{\mathbf{count}(\mathbf{w})}{|\mathbf{d}|} \quad (1)$$

где w – слово, для которого хотим получить вектор; d – текст, в котором исследуем слово; count(w) – количество вхождений слова w в текст d; |d| – общее количество слов в тексте d.

Исходя из формулы (1), можно заметить, что tf также имеет простую вероятностную интерпретацию: показывает с какой вероятностью будет выбрано слово w из текста d.

В качестве примера рассмотрим документ, который содержит 100 слов, пусть слово язык в нем встречается 7 раз, тогда: TF = 0,07. Анализируя формулу, мы понимаем, что, чем большее значение tf, тем важнее слово для конкретного текста.

Однако TF показывает только то, на сколько слово важно для конкретного документа, но, поскольку для решения задачи используется большое количество документов, необходимо также учитывать то, на сколько конкретное слово часто встречается и в остальных документах, ведь ценность слова будет на столько важна для конкретного текста, на сколько оно будет реже встречаться в остальных текстах. Исходя из этого вводится еще одно понятие – IDF (inverse document frequency – обратная частота документа).

$$\boldsymbol{idf}(\boldsymbol{w}, \boldsymbol{D}) = \boldsymbol{log}\frac{|\boldsymbol{D}|}{\{\boldsymbol{d} \in \boldsymbol{D} \mid \boldsymbol{w} \in \boldsymbol{d}\}} \quad (2)$$

где w – конкретное слово; D – коллекция текстов; |D| – количество документов в коллекции; d – текст.

Алгоритм обучения word2vec со skip-gram архитектурой

Рассмотрим общее описание работы алгоритма обучения word2vec. Для начала необходимо определить количество слов – размер словаря, который обозначим за N и

длину вложения M, пусть она будет равна 100. Создается две матрицы Embedding и Context размера N*M. Матрицы инициализируются случайными значениями. Используется размер окна равный 5. На рисунке 8 схематически показан процесс обучения.

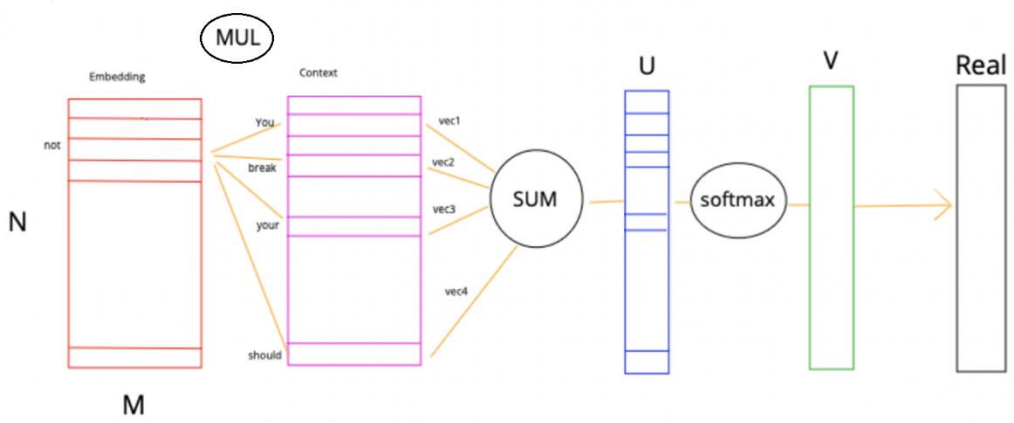

Рисунок 8 – обучение W2V

На рисунке показан процесс, в котором берется вектор центрального слова (not) из таблицы Embedding и перемножается на комбинацию векторов контекстных слов (you, should, break, your), взятых из таблицы Context, после чего получается вектор из вещественных чисел – U, но поскольку нам тяжело работать с вещественными числами, то его необходимо преобразовать в более понятную форму, а именно, в вектор вероятностей принадлежности к каждому классу, данное преобразование выполняется с помощью функции softmax, обозначенной на рисунке, как S.

После получения вероятности принадлежности к классам, имея реальный вектор принадлежности к классу, можно вычислить величину ошибки, для этого из метки класса вычитается значение сигмоиды на данном объекте – слове, на рисунке 9 показан пример вычисления ошибки.

Слово в центре	Контект	Метка класса	Сигмоида	Величина ошибки
not	you	0	0.8	-0.2
not	should	0	0.12	-0.88
not	break	1	0.6	0.4
not	your	0	0.4	-0,6

Рисунок 9 – Пример вычисления ошибки

После получения величины ошибки для каждого объекта необходимо обновить параметры модели – вектора слов, исходя из величины ошибки. Данная процедура будет выполняться с помощью градиентного спуска [6].

Заключение

В данной работе рассмотрены основные методы предобработки текстовой информации и методы формирования признаков, используемые при обработке естественного языка. Приведены примеры использования данных методов на реальных данных, полученных в интернете.

Результатом предобработки являются нормализованные по регистру и токенизированные текстовые данные. Далее эти данные преобразуются в вектора вещественных чисел – вектора признаков.

Вектора признаков могут использоваться стандартными моделями машинного обучение такими как логистическая регрессия, дерево решений, наивный байесовский классификатор [5]. Правильная предобработка и выделение значимых признаков крайне важны в подобных задачах, так как модели бывают чувствительны к качеству обучающей выборки.

Список использованных источников

1. Frank Millstein. Natural Language Processing With Python: Natural Language Processing Using NLTK – CreateSpace Independent Publishing Platform, 2018 – 120 p.
2. Zipf G.K., Human Behavior and the Principle of Least Effort. — Addison-Wesley Press, 1949. – С. 484-490. – 573 с.
3. *Фридл, Дж.* Регулярные выражения = Mastering Regular Expressions. – СПб.: «Питер», 2001. – 352 с. – (Библиотека программиста). – ISBN 5-318-00056-8.
4. Guerriero, V. (2012). «Power Law Distribution: Method of Multi-scale Inferential Statistics». *Journal of Modern Mathematics Frontier (JMMF)*, 1: 21-28.
5. Davis J., Goadrich M. The Relationship Between Precision-Recall and ROC Curves // Proc. Of 23 International Conference on Machine Learning, Pittsburgh, PA, 2006.
6. Гилл Ф., Мюррей У., Райт М. Практическая оптимизация = Practical Optimization. – М.: Мир, 1985.