ОБРАБОТКА И КОМПЬЮТЕРНЫЙ АНАЛИЗ ТЕКСТА НА ЕСТЕСТВЕННЫХ ЯЗЫКАХ

PROCESSING AND COMPUTER ANALYSIS OF NATURAL LANGUAGE TEXTS

V. Sachkov
E. Gilmutdinova
C. Matyas
D. Akimov

Summary. Natural language processing (Natural language Processing, NLP), is an important direction of development of application software, and in the future this demand will only increase.

Word Processing from natural language used to address a vast number of tasks, such as: search, annotation, classification, speech recognition, query analysis. Also used to expand the functionality of the applications, for example to simplify the user data input and text in more convenient forms, using a sequence of key operations to convert the text and extract information from it.

Keywords: NLP, NER, POS, SBD, processing, natural language, tokenization, search suggestions borders.

Сачков Валерий Евгеньевич
Аспирант, ФГБОУ ВО «Московский технологический университет»
megawatto@mail.ru
Гильмутдинова Евгения Фаритовна
Магистрант, ФГБОУ ВО «Московский технологический университет»
evgenua-1000@bk.ru
Матяш Екатерина Дмитриевна
Магистрант, ФГБОУ ВО «Московский технологический университет»
katya_matyash@mail.ru
Акимов Дмитрий Александрович
К.т.н., ФГБОУ ВО «Московский технологический университет»
akimovdmitri@gmail.com

Аннотация. Обработка естественного языка (Natural language Processing, NLP), представляет собой важное направление разработки прикладного программного обеспечения, и в будущем эта потребность будет только возрастать.

Обработка текстов на естественном языке используется для решения обширного числа задач, таких как: поиск, аннотирование, классификация, распознавание речи, анализ запросов. Также применяется для расширения функциональной возможности приложений, например для упрощения ввода пользователем исходных данных и преобразование текста в более удобные формы, используя при этом последовательность ключевых операций для преобразования текста и извлечения из него информации.

Ключевые слова: NLP, NER, POS, SBD, Обработка, естественные языки, токенизация, поиск границ предложения.

Введение

Обработка естественного языка (Natural language Processing, NLP) — это обширная область ИТ, связанная с использованием компьютеров для анализа естественных языков, к которым относятся такие дисциплины, как распознавание, обработка, реферирование, аннотирование, категоризация и.т.д. Существует большое разнообразие задач обработки естественного языка:

1. Поиск фрагментов текста — разделение материала на различные элементы разных типов: слова, предложения, абзацы и.т.д.
2. Поиск предложений (Sentence Boundary Disambiguation, SBD) — определение границ предложения.
3. Поиск именованных объектов (Named entity recognition, NER) — механизм поиска адресов, названий, имен, дат, или любых других именованных сущностей.
4. Определение частей речи (Parts of speech, POS) — классификация элементов текста на уровне предложения. Предложение может быть разделено на отдельные слова и словосочетания по таким категориям, как существительные, глаголы, наречия, предлоги и.т. Д.
5. Классификация текстов и документов — цель данной классификации в присвоении меток фрагментам, найденным в текстах и документах.
6. Выделение взаимоотношений — выявление связей между словами или словосочетаниями, для построения семантического дерева.

Несмотря на большое количество разнообразных задач анализа текста, можно выделить базовый алгоритм,

Таблица 1. Инструменты обработки NLP

API	URL
LingPape	http://alias-i.com/lingpipe/
Apache OpenNLP	http://opennlp.apach.org/
Stanford Parser	http://nlp.stanford.edu/software
UIMA	http://uima.apache.org/
Mallet	http://mallet/cs/umass.edu/

Таблица 2. Список пробельных символов

Представление в Unicode	Символ	Обозначение	Расшифровка
\t	Табуляция	HT	Horizontal tabulation
\v	Вертикальная табуляция	VT	Vertical tabulation
\r	Возврат каретки	CR	Carriage return
\n	Перевод строки	LF	Line feed
\f	Конец страницы	FF	Form feed
\e	Escape-символ	ESC	Escape character
\b	Забой	BS	Backspace

Листинг 1.

```
      private static String getHTML(String urlToRead) {
      URL url;
      HttpURLConnection conn;
      BufferedReader rd;
      String line;
      String response = «»;
      String typeRequest = null;
      List<String> postList;
      String result = null;
      try {
      Integer id = new Integer(urlToRead);
      typeRequest = «&owner_id=»;
      } catch (Exception e) {
      typeRequest = «&domain=»;
      }
      String pattern = «http://api.vk.com/method/wall.get?v=5.37» + typeRequest +
urlToRead + «&filter=owner&count=5»;
      try {
      url = new URL(pattern);
      conn = (HttpURLConnection) url.openConnection();
      conn.setRequestMethod(«GET»);
      rd = new BufferedReader(new InputStreamReader(conn.getInputStream()));
      while ((line = rd.readLine())!= null) {
      response += new String(line.getBytes(), «UTF-8»);
      }
      rd.close();
      } catch (Exception e) {
      e.printStackTrace();
      }
      postList = parseJson(response);
      for (String post: postList) {
      result += post;
      }
      return result;
      }
```

Листинг 2.

```
private static List<String> parseJson(String json) {
List<String> stringList = new LinkedList<String>();
JSONObject obj = new JSONObject(json);
JSONArray response = obj.getJSONObject(«response»).getJSONArray(«items»);
System.out.println(«read post from vk.com»);
for (int i = 0; i < response.length(); i++) {
JSONObject itemList = response.getJSONObject(i);
try {
if (itemList.getString(«text»).length() > 0) {
String obj2 = itemList.getString(«text»);
stringList.add(obj2);
} else {
JSONArray obj2 = itemList.getJSONArray(«copy_history»);
for (int j = 0; j < obj2.length(); j++) {
JSONObject postText = (JSONObject) obj2.get(j);
stringList.add(postText.getString(«text»));
}
}
} catch (Exception e) {
e.printStackTrace();
}
}
return stringList;
}
```

Листинг 3.

```
public static void main(String[] args) {
List<String> tokensList = new LinkedList<>();
WhitespaceTokenizer whitespaceTokenizer = WhitespaceTokenizer.INSTANCE;
String[] tokens = whitespaceTokenizer.tokenize(getHTML(«8486734»));
Collections.addAll(tokensList, tokens);
System.out.println(tokensList);
}
```

применяемый в большинстве методов обработки текста с применением компьютера:

1. Разделение текста на фрагменты
2. Определение границ предложений
3. Выделение отношений между элементами

На данный момент уже существует множество инструментов, библиотек и алгоритмов для обработки и анализа текста, некоторые из них приведены в таблице № 1.

Однако, не смотря на кажущиеся простоту, задачу обработки текста на естественном языке, нельзя назвать простой.

Разделение текста

Самой первой задачей при обработке текста, является поиск и разделение текста на отдельные, мелкие, удобные для работы фрагменты или *токенизация*. Токенизация — это процесс разделения текста на более мелкие фрагменты, для многих текстов это слова. Разделение чаще всего происходит по специальным *символам-разделителям (*Пример: таблица № 2).

Для примера, попробуем токенезировать текст «постов» из социальной сети «ВКонтакте», с помощью библиотеки «Apache OpenNLP».

Для начала нам нужно написать парсер, который будет считывать текст из постов в социальной сети (ограничимся 5 последними «постами»).

Воспользовавшись их API, напишем следующий код на языке программирования Java.

Данный код будет возвращать текст 5 последних «постов».

Соответственно так как «ВКонтакте» возвращает ответ в виде JSON, нам нужно привести это сообщение в удобный для обработки вид. Для этого напишем следующий код в Листинге 2.

После загрузки и предварительно подготовки текста, воспользуемся стандартным классом WhitespaceTokenizer токенизации из библиотеки OpenNLP. Это самый простой алгоритм токенизации, который разбивает текст по пробельным символам.

После работы программы, проанализируем результаты:

- =)
- =D
- =P
- To
- чувство
- когда…
- 😁Чат
- боты
- рулят
- 😁Побывал
- сегодня
- на
- докладе
- разработчиков
- Kotlin
- довольно
- интересная
- задумка,
- есть
- перспективы
- возможно
- стоит
- присмотреться
- к
- технологии
- =)
- https://kotlinlang.org/Я
- не
- буду
- особенно
- оригинальный
- поэтому
- скромно
- поздравлю
- всех
- девушек
- открыткой)))

Такой метод не очень подходит, для обработки сложного не структурированного текста. В токены попало «шумовые слова», стоп-слова (предлоги, союзы и.т.д.), а так же спец. символы.

Но не всегда процесс токенизации, является легкой задачей, сложностью этого процесса можно выделить несколько факторов:

- Язык — в каждом языке есть свои грамматические правила. В большинстве языков разделителями слов являются пробельные символы. Однако в китайском языке пробелы не используются, там нужен другой набор разделителей.
- Формат текста — хорошим примером может послужить формат HTML где текст находится в разметках тегов, которое сильно усложняет задачу выделения текста из тегов.
- Шумовые слова — люди не роботы и могут совершать ошибки или даже специально писать абсолютно бессмысленные слова или случайные наборы символов, которые были бы крайне не желательны в системе обработке текста, так как не несут никакого логического смысла и создают погрешности и ошибки. Так же в текстах распространены «Стоп-слова» — это наиболее часто встречающие слова, которые не несут никакой смысловой нагрузки, например союзы «а», «но» и.т.д.
- Регистр символов — в некоторых случаях может играть важную роль в определении имен, мест, названий и.т.д.

Результаты токенизации можно использовать для проверки правописания, определения частей речи, стеммизации, но главное назначение это подготовка набора данных для дальнейшей обработки.

Определение границ предложений

Следующим шагов в обработке текста является определение границ предложений и абзацев (Sentence Boundary Disambiguation, (SBD)).

Процесс поиска границ предложения очень часто зависит от конкретного языка. Общие методы поиска предполагают использование набора правил и обученной модели с помощью одного из методов машинного обучения. Самые простые правила поиска:

- Предложение завершается точкой, восклицательным или вопросительным знаком
- Точке не предшествует, какое либо сокращение или цифровой символ

Но при таком подходе возникают трудности, когда в тексте встречаются сокращения, аббревиатура или специальные символы, например многоточие, которые не позволяют однозначно определить, является ли точка окончанием предложения. Также вызывают сложности такие элементы как цитаты, в которые могут, входить множество предложений, разделенных всеми знаками препинания и точками.

Пример сложного предложения:

«Имущему дастся, а у неимущего отнимется», помнишь? Она — неимущий: за что? не знаю; в ней нет, может быть, эгоизма,— я знаю, но у неё отнимется, и всё отнялось. Мне её ужасно жалко иногда; я ужасно желала прежде, чтобы Nicolas женился на ней; но я всегда как бы предчувствовала, что этого не будет. Она пустоцвет, знаешь, как на клубнике? Иногда мне её жалко, а иногда я думаю, что она не чувствует этого, как чувствовали бы мы. (Л. Толстой «Война и мир»)

Более современный подход, ищет границы предложения по нескольким наборам токенов и флагов:

- Набор завершающих токенов — токены, которые могут завершить предложение, например «.»
- Набор невозможных предпоследних завершающих токенов — токены, которые не могут находиться перед завершающим токеном
- Набор невозможных начальных токенов — токены, которые не могут находиться в начале предложения
- Парность скобок — флаг определяющий, что предложение не может завершиться без закрывающей скобки
- Принудительная установка границ — флаг показывающий, что данный токен является окончанием предложения, даже если он не входит в набор «завершающих токенов»

Выделение отношений между элементами. Поиск именованных объектов

А теперь используя подготовленные данные, можно, наконец-то сделать что-то полезное, например, найти в тексте именованные сущности (Named entity recognition, NER)

Именованные сущности — это имена существительные, которые обозначают конкретные экземпляры объектов, имена, названия и.т.д. Во многих ситуациях, также полезно кроме распознавания имен, мест и названий, узнать дату, денежную единицу, числа указанное в тексте и другие сущности (рис 1).

Идентификация имен людей, названий организаций, мест и других именованных сущностей позволяет уяснить характер сущности и предпринять соответствующие действия. Например, располагая этой информацией, мы можем предложить дополнительные сведенья о сущностях, рекомендовать сопутствующие материалы и в конце концов повысить интерес к нашему приложению или сайту. Допустим, человек читает статью на новостном сайте, а сайт ему предлагает ссылки на соответствующие темы или людей, о которых ведется речь, человек, переходит по ссылке и дальше снова получает тематически ссылки на статью по этой тематике.

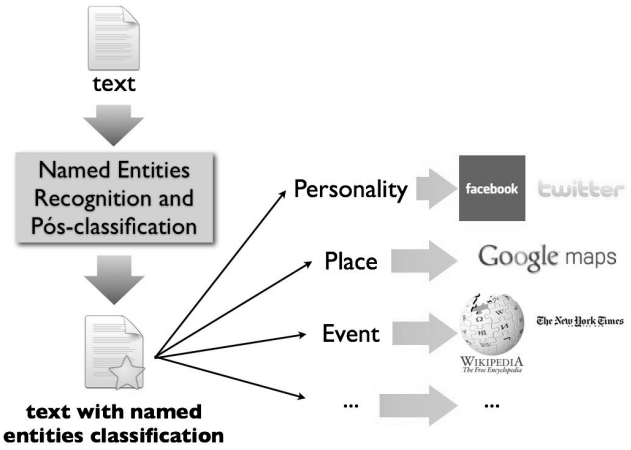

Рис. 1. Именные сущности

Одним из первых решений данного подхода были системы, основанные на большом количестве правил *регулярных выражений* (RegEx), суть ее заключалась в следующем: используя синтаксис регулярных выражений, создавались специальный шаблон, по которому ищутся совпадения в тексте например шаблон для поиска email адреса в тексте:

```
[a-zA-Z1-9\-\._]+@[a-z1-9]+(.[a-
z1-9]+){1,}
```

Соответствует любому стандартному email адресу.

```
(http|ftp|https):\\/\\/
[\\w\\-_]+(\\.[\\w\\-_]+)+([\\w\\-
\\.,@?^=%&amp;:/~\\+#]*[\\w\\-
\\@?^=%&amp;/~\\+#])?
```

Соответствует любому стандартному URI.

```
(\\:\\w+\\:|\\<[\\/\\\\]?3|[\\
(\\)\\\\\\D|\\*\\$][\\-
\\^]?[\\:\\;\\=]|[\\:\\;\\=B8][\\-
\\^]?[3DOPp\\@\\$\\*\\\\\\)\\(\\/\\|])
(?=\\s|[\\!\\.\\?]|$)
```

Соответствует любому стандартному «смайлу».

Используя данный подход, приведем пример поиска именных сущностей, таких как URI адреса и «смайлов» по нашим токенам полученных в предыдущем разделе. Листинг 4.

Листинг 4.

```
      private static String NER(String text) {
      String chunkUri =
      «(http|ftp|https):\\/\\/[\\w\\-_]+(\\.[\\w\\-_]+)+([\\w\\-
\\.,@?^=%&amp;:/~\\+#]*[\\w\\-\\@?^=%&amp;/~\\+#])?»;
      String chunkSmile = «(\\:\\w+\\:|\\<[\\/\\\\]?3|[\\(\\)\\\\\\D|\\*\\$][\\-
\\^]?[\\:\\;\\=]|[\\:\\;\\=B8][\\-\\^]?[3DOPp\\@\\$\\*\\\\\\)\\(\\/\\|])(?=\\
s|[\\!\\.\\?]|$)»;
      Pattern patternUri = Pattern.compile(chunkUri);
      Pattern patternName = Pattern.compile(chunkSmile);
      Matcher matcherUri = patternUri.matcher(text);
      Matcher matcherSmile = patternName.matcher(text);
      if (matcherUri.find()) {
      return («URI: « + matcherUri.group());
      }
      if (matcherSmile.find()) {
      return («Name: « + matcherSmile.group());
      }
      return null;
      }
```

Результат данного кода выведет следующие результаты:

```
       [Name: =), Name: =P, Name: =D,
Name: =), URI: https://kotlinlang.org/]
```

Как видим, алгоритм смог без ошибочно найти все именные сущности, которые были у нас в токенах.

Но такой подход утратил позиции из-за слишком сложного обслуживания системы, так как шаблонов было много, очень сложный переход на другие предметные области, переход на другие технологии требовал переформатирования шаблонов, плохая работа со словосочетаниями и множество других проблем.

Современный подход это использование статических классификаторов на базе обученных моделей машинного обучения. Где уже модель по количеству признаков сама определяет, является ли объект именованной сущностью, такой подход является более гибким и имеется довольно много способов обучения классификаторов. Но у данного подхода есть и недостатки. Чтобы обучить модель, нужно предварительно человеком подготовить, специально размеченный набор документов, в среднем в таких документах около 30000 слов.

Не зависимо, какой классификатор NER применяется, современные системы способны распознавать с точность около 90%, при этом легко модифицируется и подстраивается под разные предметные области, а также постепенно совершенствуется, по мере накопления и обработки данных.

Определение тональной окраски текста

Тональность, это текстовая категория, отражающая эмоциональную установку автора или его психологическую позицию к излагаемому или адресату и ситуации, т.е. анализ данной категории, позволяет выявлять и извлекать из текста субъективную информацию.

Для примера мы опробуем определить эмоциональное состояние человека по его 5 последним постав в социальной сети «ВКонтакте».

Решим данную задачу по следующему алгоритму:
1. Подготовим обучающую выборку
2. Тренировка классификатора по обучающей выборке
3. Используем метод токенизации, для получения токенов из постов.
4. Выделим именные сущности, такие как «смайлы».
5. С помощью обученной модели определим настроение человека.

Обучающая выборка состоит из 12 «постов», из которых 7 положительных, 3 негативных и 2 нейтральных. Каждый из них обладает следующими свойствами.

В качестве метода классификатора, используем «наивный байесовский классификатор». Данный метод позволяет прогнозировать класс при определенном наборе характеристик, используя вероятность их появления.

Таблица 3. Обучающая выборка

Настроение	=)	=(	=\	=D	:(
Позитивное	20	1	4	5	1
Негативное	1	10	4	1	3
Нейтральные	5	2	4	2	2

Листинг 5.

```
private static Map<String, Map<String, Double>> ton() {
double totalDoc = 12;
String[] tonClass = {«pos», «neg», «another»};
Integer[][] train = {{20, 1, 4, 5, 1}, {1, 10, 4, 1, 3}, {5, 2, 4, 2, 2}};
double posTotal = (7 / totalDoc);
double negTotal = 3 / totalDoc;
double anotherTotal = 2 / totalDoc;
classConst.put(«pos», posTotal);
classConst.put(«neg», negTotal);
classConst.put(«another», anotherTotal);
Map<String, Map<String, Double>> model = new HashMap<>();
for (int c = 0; c < tonClass.length; c++) {
Map<String, Double> attr = new HashMap<>();
for (int a = 0; a < tonAttr.length; a++) {
double avg = 0;
for (int i = 0; i < train.length; i++) {
avg += train[i][a];
}
attr.put(tonAttr[a], (train[c][a] / avg));
}
model.put(tonClass[c], attr);
}
System.out.println(model);
return model;
}
```

Упрощенная формула для классификации выглядит примерно так:

$$P(\text{Класс A} \mid \text{Свойство 1, Свойство 2}) = \frac{P(\text{Свойство 1} \mid \text{Класс A}) * P(\text{Свойство 2} \mid \text{Класс A}) * P(\text{Класс A})}{P(\text{Свойство 1}) * P(\text{Свойство 2})}$$

В нашем случае мы использовали следующие формулы:

1. $P(\text{класс}) = \frac{\sum \text{кол-во постов класс}}{\sum \text{общ.кол-во постов}}$

2. $P(\text{свойство}|\text{класс}) = \frac{\text{частота вхождения свойства в класс}}{\sum \text{всех вхождений свойста}}$

3. $P(\text{класс}|\prod \text{свойств} * P\ (\text{класс})) = \prod \text{свойств} * P\ (\text{класс})$

4. $\text{Класс} = argmax\, P(\text{класс}) * \prod \text{свойств}$

Листинг 6.

```
private static void classification(Map<String, Map<String, Double>> model,
List<String> tokens) {
String maxClass = null;
double maxVer = 0;
for (Map.Entry classTon: model.entrySet()) {
Double ver = null;
Map<String, Double> cA = (Map<String, Double>) classTon.getValue();
for (String token: tokens) {
if (ver == null) {
ver = cA.get(token);
} else {
if (cA.get(token)!= null) {
ver *= cA.get(token);
}
}
}
ver *= classConst.get(classTon.getKey());
if (ver > maxVer) {
maxVer = ver;
maxClass = (String) classTon.getKey();
}
}
System.out.println(«final:» + maxClass + « = « + maxVer + «%»);
}
```

В первую очередь напишем код, который тренирует нашу модель (листинг 5).

Далее напишем алгоритм классификации на базе нашей обученной модели (листинг 6).

По окончанию работы алгоритма, из наших обработанных данных, модель классифицировала настроение автора, как «позитивное», что является верным. Из всех классов самая большая вероятность оказалась у класса «pos».

final: neg = 0.003698224852071%
final: pos = 0.3451676528599606%
final: another = 0.00616370808678501%

Поскольку обучающая выборка довольно маленькая и служит лишь демонстрационным примером. Поэтому она показывает довольно низкий процент принадлежности к классу. Но даже этого вполне достаточно для решения простых задач, по определению тональности текста.

ЛИТЕРАТУРА

1. Грант С. Ингерсолл, Томас С. Мортон, Эндрю Л. Фэррис. Обработка неструктурированных текстов. Поиск, организация и манипулирование. / Пер. с англ. Слинкин А. А.— М.: ДМК Пресс, 2015.— 414 с.
2. Риз Р. Обработка естественного языка на Java / пер. с англ. Снастина А. В.— М.: — ДМК Пресс, 2016.— 264 с.
3. Луис Педро Коэльо, Вилли Ричард. Построение систем машинного обучения на языке Python. 2-е издание /пер. с англ. Сликин А. А.— М.: ДМК Пресс, 2016.— 302 с.
4. Представление символов в регулярных выражениях [Электронный ресурс] — https://ru.wikipedia.org/wiki/Представление_символов_в_регулярных_выражениях — статья в интернете.
5. Поиск именованных объектов [Электронный ресурс] — https://en.wikipedia.org/wiki/Named-entity_recognition — статья в интернете.
6. Алгоритмы интеллектуального анализа данных [Электронный ресурс] — https://tproger.ru/translations/top-10-data-mining-algorithms — статья в интернете.

© Сачков Валерий Евгеньевич (megawatto@mail.ru), Гильмутдинова Евгения Фаритовна (evgenua-1000@bk.ru),
Матяш Екатерина Дмитриевна (katya_matyash@mail.ru), Акимов Дмитрий Александрович (akimovdmitri@gmail.com)
Журнал «Современная наука: актуальные проблемы теории и практики»