

## АЛГОРИТМЫ РАСПОЗНАВАНИЯ СХОЖЕСТИ ТЕКСТОВ

Нахождение сходства между парами текстов является серьезной проблемой для автоматической обработки текстов (АОТ). Такая проблема возникает в различных задачах АОТ, таких как машинный перевод, автоматическое построение рефератов, определение авторства, обнаружение академического плагиата, поиска информации и многие другие, в которых нужно определить и измерить степень подобия между двумя заданными текстами.

Наиболее известными алгоритмами распознавания схожести текстов являются коэффициент Жаккара и алгоритм шинглов.

**Коэффициент Жаккара.** Иначе называется мерой Жаккара или коэффициентом сходства.

Коэффициент сходства Жаккара вычисляет количество уникальных терминов, совместно используемых между двумя текстами, и рассчитывается по формуле:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|},$$

где  $A$  – количество уникальных терминов первого текста,  
 $B$  – количество уникальных терминов второго текста.

Перед сравнением оба текста нужно преобразовать – очистить от ненужных нам знаков и слов, которые не несут смысла при сравнении. На этом шаге удаляются знаки препинания и табуляции, отступы, лишние пробелы, слова приводятся к одному регистру. Предлоги, союзы и другие служебные части речи удаляются. Следующий шаг – лемматизация. Это метод морфологического анализа, который сводится к приведению словоформы к ее первоначальной словарной форме. Все прилагательные приводятся к мужскому

роду единственного числа в именительном падеже, а глаголы – к инфинитиву. Затем стемминг – это поиск основы слова, учитывающий морфологию исходного слова. Решим эту задачу с помощью языка программирования Python.

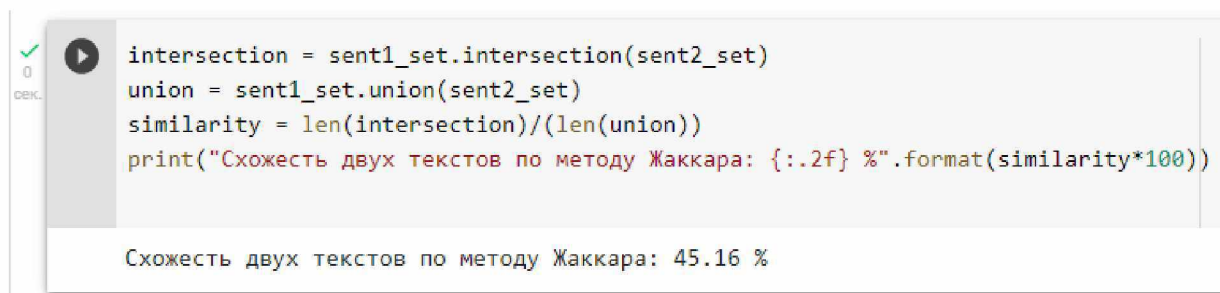
Python существует так долго, что разработчики смогли сделать специальные библиотеки практически для любых целей. Предобработка текста: удаление стоп-слов, стемминг и лемматизация выполняется с помощью Python-библиотек: `re`, `rumorphy2` и `NLTK`.

После предобработки тексты сравниваются, и выводится процент их схожести.

Проверим, как работает алгоритм, сравним два текста (рисунок).

*Текст 1. Информационная технология – совокупность методов, производственных процессов и программно-технических средств, объединенных в технологический комплекс, обеспечивающий сбор, создание, хранение, накопление, обработку, поиск, вывод, копирование, передачу, распространение, удаление (уничтожение) и защиту информации.*

*Текст 2. Информационная технология – совокупность процессов, методов осуществления поиска, получения, передачи, сбора, обработки, накопления, хранения, распространения и (или) предоставления информации, а также пользования информацией и защиты информации.*



```
intersection = sent1_set.intersection(sent2_set)
union = sent1_set.union(sent2_set)
similarity = len(intersection)/(len(union))
print("Схожесть двух текстов по методу Жаккара: {:.2f} %".format(similarity*100))
```

Схожесть двух текстов по методу Жаккара: 45.16 %

Пример работы алгоритма

**Алгоритм шинглов** был разработан для поиска заимствований частей одного текста другим текстом. Реализация алгоритма шинглов подразумевает несколько этапов:

- 1) предобработка (канонизация) текста;
- 2) разбиение текстов на шинглы;
- 3) вычисление контрольных сумм;
- 4) поиск одинаковых шинглов;
- 5) вывод результата.

На первом этапе, как и в предыдущем алгоритме, нужно получить текст, готовый для сравнения, то есть выполнить его предобработку.

Шинглы представляют собой выделенные из текста подпоследовательности слов, следующих внахлест, подобно чешуйкам (от английского *shingles* – ‘чешуйки’). Для выделения шингла необходимо из сравниваемых

текстов выделить  $N$  слов, идущих друг за другом. Таким образом, разбивая текст на подпоследовательности, мы получим набор шинглов. Число шинглов для документа со сдвигом шингла в  $M$  слов будет составлять:

$$K = L - N + M,$$

где  $K$  – число шинглов в документе,

$N$  – длина шингла,

$M$  – сдвиг шингла ( $M < N$ ),  $L$  – число слов в документе.

Способ формирования шинглов и количество слов или символов в шингле, а также сдвиг шингла (на сколько слов или знаков сдвигается последующая подстрока) сильно влияет на точность результата. При определении размерности подстроки выбор зависит от вычислительной мощности, объемов памяти и требуемой точности результатов. Шинглы могут быть на любое количество слов, чем шингл короче, тем точнее будет результат проверки.

Вычисление контрольных сумм (контрольная сумма, иначе сигнатура – многозначный термин, в данном случае – уникальная цифровая характеристика текста) осуществляется с помощью специальных функций, называемых хеш-функциями. В результате каждому шинглу ставится в соответствие некоторое уникальное число. Для вычисления контрольных сумм применяются разновидности хеш-функций (например, SHA1, MD5, CRC32 и т. д.). При составлении программы на языке программирования Python, можно использовать алгоритм хэширования CRC32, подключив библиотеку `binascii`, которая содержит в себе нужный метод.

Таким образом, после канонизации текста и деления на шинглы (подстроки) вычисляется контрольная сумма для двух текстов.

Далее осуществляется непосредственное сравнение полученных контрольных сумм текстов и подсчитывается процент их совпадения.

Представленный алгоритм можно дорабатывать до бесконечности, увеличивая его производительность и точность сравнения. Для увеличения производительности при обработке больших объемов текста можно сравнивать не все полученные контрольные суммы, а только те, которые, например, делятся на 25, или любое целое число в пределах от 10 до 40.

Существуют модифицированные версии «Алгоритма шинглов» – «Алгоритм супершинглов» и «Алгоритм мегашинглов».