Министерство образования и науки Российской Федерации

Московский физико-технический институт (государственный университет)

Физтех-школа прикладной математики и информатики

Кафедра теоретической и прикладной информатики

Выпускная квалификационная работа бакалавра

Морфологический анализ текста

Автор:

Астанова Камилла Сайфидиновна

группа Б05-905а

Научный руководитель:

Левицкий Михаил Сергеевич

Москва 2023

Аннотация

Содержание

Обозначения и сокращения

CL – Computational Linguistics

NLP – Natural Language Processing

FSM – Finite-state Machine

FST – Finite-state Transducer

DAFSA – Deterministic Acyclic Finite State Automaton

DAWG – Deterministic Acyclic Word Graph

НКРЯ – Национальный Корпус Русского Языка

1 Введение

В современном мире существует множество языковых средств и инструментариев, помогающих облегчить и улучшить процесс обработки естественного языка. Среди них наибольшую популярность имеют морфологические анализаторы, которые используются в широком диапазоне задач - от поискового анализа до создания чат-ботов.

В данной работе будет проведено сравнение двух наиболее известных морфологических анализаторов - pymorphy2 и pymystem3. Цель исследования - определить, какой из них более эффективен в решении задач морфологического анализа русского языка.

Для достижения цели был проведен обзор методологии работы каждого анализатора, исследование качества анализа и скорости работы при различных условиях. Полученные результаты позволяют сформулировать заключение о преимуществах и недостатках каждого анализатора и определить, который из них лучше подходит для конкретных задач.

2 Постановка задачи

2.1 Цель и задачи

В экспериментальной части данного дипломного исследования было проведено сравнение двух популярных морфологических анализаторов - pymorphy2 и pymystem3. Наличие нескольких инструментов для морфологического анализа текстов обусловлено необходимостью создания эффективных и точных систем обработки естественного языка. В целях повышения качества и точности морфологического анализа важно выявлять преимущества и недостатки различных инструментов.

Цель проведенного исследования - выявить эффективность и точность морфологического анализатора pymorphy2 и pymystem3. Результаты сравнения могут быть использованы для выбора наиболее подходящего инструмента для конкретных целей обработки естественного языка.

Целью данного исследования является не только выявление эффективности инструментов, но и сравнение различий в реализациях морфологического анализа этих инструментов. Кроме того, данное исследование может быть интересно для студентов и коллег, занимающихся обработкой естественного языка и выбирающих между различными морфологическими анализаторами.

2.2 Определения

Инсайдер

2.3 Наборы данных

3 Обзор литературы

The state-of-the-art algorithms for morphological analysis are based on the Finite State Transducers of different flavours ([kimo1983], [kartunen1992], [mohri2000], [daciuk2000]), where phonological rules are described by hand and each word from the dictionary is assigned its corresponding (inflectional/derivational) model. These transducers allow both analysis and synthesis of the word forms; they are very fast and robust. Unfortunately number of unknown lexical items brought from the web with every crawl are very big and can be estimated by Heaps law. This means that the dictionary grows sub linear with grow of index. One approach here is simply to ignore all the words that are not from main dictionary [www.aport.ru] hoping that the coverage will suffice. Recently the problem of the morphological analysis of the unknown words draws more attention mostly from the success in unsupervised learning of morphology from the corpora itself. [Jackuemin1997] analysed statistics of k-n conflations (where k – length of the hypothesized suffix, usually 3 letters, and n – the width of the context window – usually 2 words) and obtained decent results compared to Lovins and Porter famous hand-made stemmers ([lovins1968], [porter1980]). [gaussier1999] used similar approach (p-similarity – the number of common letters between two words) to learn derivational morphological families and even derivational trees. The more generalized approaches of corpus suffix analysis are presented in [goldsmith2001] and [snover2002]. They use MDL (Minimum Description Length) principle, known from theory of information, to produce the set of stems, signatures (paradigms) and affixes that can be encoded in the minimal number of bits [goldsmith2001] or that are most probable (Brent) thus making morphological description of corpus minimal in some sense. Linguistica [goldsmith2001] is also a publicly available tool that we will use later in this paper. Another promising approach [schone2001] makes use of word contexts to distillate hypothesized models obtained from pure suffixal analysis. It uses the primitive of PPMV — pair of potential morphological variants and measures semantic similarity between them, which is understood as the normalized correlation between their vectors in the latent semantic space (which in fact is word-to-wordneighbours matrix factorised with the help of LSI).

4 Теоретическая часть

4.1 Вычислительная морфология

4.1.1 Вступление

Ежедневно появляется огромное количество текстов, которые необходимо проанализировать. Кроме того, большое количество информации в базах данных не может быть интерпретировано без участия человека. Все это явилось причиной стремительного развития **компьютерной лингвистики (Computational Linguistics)**. Это направление научных исследований, которое занимается созданием алгоритмов и программ для **обработки естественного языка (Natural Language Processing)** компьютерами. Она сочетает в себе знания из лингвистики, математики, статистики и информатики. Целью компьютерной лингвистики является разработка систем, способных понимать и генерировать естественный язык, а также извлекать полезную информацию из текстовых данных.

Свои истоки наука берет в 1950-х годах, когда известный американский лингвист Ноам Хомский проводил исследования по формализации структуры естественного языка[1] и пробные эксперименты по машинному переводу. А уже в январе 1954 года в Джорджтаунском университете (США) был проведен первый в мире публичный эксперимент по машинному переводу, где инженерам удалось полностью автоматически перевести более 60 предложений с русского языка на английский (Georgetown-IBM[2]). Это событие можно считать зарождением компьютерной лингвистики.

Популярность CL продолжает расти, так как она является ключевым элементом для создания интеллектуальных систем. В будущем, по прогнозам экспертов, компьютерная лингвистика будет играть еще более значимую роль в создании автоматизированных систем, которые будут способны понимать и обрабатывать естественный язык.

Так как в компьютерно-лингвистической обработке текстов используются естественные языки, необходимо иметь базовые знания в общей лингвистике. Она включает в себя несколько областей: **фонологию** (изучение звуковой структуры языка), **морфологию** (исследование внутренней и внешней формы слов), синтаксис (анализ структуры предложений), **семантику и прагматику** (изучение значения слов и их использования в определенных контекстах) и **лексикографию** (описание словарного состава языка).

Рис. 1: Разделы общей лингвистики

4.1.2 Приложения компьютерной лингвистики

ПЕРЕПИСАТЬ!!!!! НАПИСАЛА ЕРЕСЬ НЕСТРУКТУРИРОВАННУЮ

Опишем наиболее актуальные задачи прикладные задачи, так как полный список применения алгоритмов обработки речи сложно охватить.

- **Машинный перевод** - это одна из наиболее актуальных и важных областей компьютерной лингвистики. К концу 1950-х годов была создана первая компьютерная программа, способная переводить тексты целиком на основе словарей и правил. В 1960-х годах появились первые системы машинного перевода на основе статистических методов, параллельно развивались экспертные системы, основывавшиеся на знаниях профессиональных переводчиков и лингвистических правилах. Разрабатывались гибридные системы, которые позволили достичь более высокого качества перевода. Наконец, в 2014 году была создана первая нейронная сеть для машинного перевода, что дало возможность повысить точность и эффективность перевода благодаря анализу контекста и последовательности слов в предложении. Распространенными примерами машинного перевода являются Google Translate, Yandex.Translate.

- Для осуществления полнотекстового поиска документов в больших базах текстовых документов необходимо произвести **индексирование** текстов, что требует предварительной лингвистической обработки и создания специальных индексных структур. Одной из наиболее распространенных моделей информационного поиска является векторная модель, при которой запрос представляется в виде набора слов, а релевантные документы определяются на основе сходства между векто-

ром запроса и вектором слов документа. Для более точной выдачи релевантных документов современные интернет-поисковики используют сложные процедуры ранжирования. В настоящее время одним из актуальных направлений исследований в области информационного поиска является разработка многоязыковых систем поиска.

- **Реферирование текста** - одно из приложений CL, которое позволяет автоматически сократить объем текста и извлечь основную информацию. Этот метод активно используется в в информационном поиске, анализе новостей и научных статейд и др. Для реферирования текста часто используются нейронные сети, статистические модели и другие инструменты компьютерной лингвистики. Одним из главных преимуществ реферирования текста является возможность быстрого поиска информации в больших коллекциях документов, что повышает эффективность работы и экономит время. Схожая задача ставится при **аннотировании текста** - выделения ключевых тем, идей и фактов из документа.

- **Кластеризация текста** относится к задаче разбиения набора документов на несколько групп (кластеров) таким образом, чтобы документы в одном кластере были более похожи друг на друга, чем на документы из других кластеров. Эта задача может быть полезна для организации больших объемов информации или анализа тематической структуры текстов. **Классификация текста**, с другой стороны, предполагает определение принадлежности документа к заранее заданным категориям на основе его содержания. Это может использоваться для автоматического анализа тональности отзывов, определения авторства документов и многих других приложений.

- **Question Answering** (QA) позволяет пользователям задавать вопросы на естественном языке и получать ответы. Системы QA могут работать с различными типами вопросов, например, основанными на фактах или требующими анализа и интерпретации текста.

- Анализ тональности и эмоциональной окраски текста для выявления мнений и отношения людей тоже одно из приложений компьютерной лингвистики. **Opinion Mining&Sentiment Analysis** - технологии обработки естественного языка, которые позволяет определять и анализировать мнения людей в текстовых сообщениях: отзывы, комментарии и сообщения в социальных сетях, - и в более широких

тематических контекстах (статьи и тексты про социальную и экономическую обстановку, политику или спорт и др..)

4.2 Методы морфологического анализа текста

4.2.1 Основные понятия

Использование в научных текстах термина "слово"может добавить ряд дополнительных исключений из-за многозначности данного определения, поэтому часто используются более строгие понятия. Начнем с определения **токена**, как минимальной лингвистической единицей текста (речи), обладающей смыслом (слово, пунктуация, число) и полученной из текста после исключения разделителей (пунктуации и знаков препинания). Каждый токен имеет свою начальную форму или **лемму**, от которой образуются все остальные формы слова путём **флексии** - изменения начальной формы в соответствии с **грамматическими характеристиками**, такими как часть речи, число, род и т. д. Совокупность этих характеристик получила краткое название **тег**. **Словоформа** представляет собой группу (кортеж), состоящую из токена, его начальной формы и грамматических параметров. Тогда под **лексемой** понимается множество всех словоформ, связанных с данной начальной формой.

В **изолирующих** языках слово обычно представлено корнем, а взаимоотношения между словами передаются синтетически (интонация, порядок слов и т.д.). В **агглютинативных** языках структура слова определяется большим количеством аффиксов, прибавляемых к неизменяемой основе слова. Аффиксы в агглютинативных языках располагаются в установленном порядке и имеют строго определенное значение. Турецкий и казахский языки являются примерами агглютинативных языков. **Флективные** языки характеризуются многозначностью грамматических морфем и крепкой спаянностью всех морфем в слове. Создание морфологических анализаторов для флективных языков представляет особую сложность, в то время как для агглютинативных языков - задача более легко решаемая.

4.2.2 Словарный подход

Один из традиционных подходов к морфологическому анализу русского языка заключается в создании словарной морфологии на основе словаря. Построение словаря обычно имеет два подхода и их видоизменения.

В простейшем случае создается словарь всех словоформ: представляется в виде

таблицы с записями <словоформа><начальная форма><грамматические характеристики>. Тогда задача морфологического разбора сводится к поиску конкретной записи в словаре. Данный подход актуален для высокофлективных языков.

Более предпочтителен словарь основ. В данном случае структура представляет собой записи о всевозможных основах словообразовательных классов и связанный с ним словарь возможных флексий. Процедура разбора при этом не сильно усложняется: последовательно отсекаются возможные окончания от 0 до n, разбивая токен на две составные части. Для полученной флексии отдельно ищется словоизменительный класс по словарю окончаний. Аналогичная процедура производится с основой (только теперь она ищется в соответствующем словаре) и по найденной записи так же восстанавливается словоизменительный класс. В случае совпадения классов, токену сопоставляются грамматические характеристики и лемма.

Среди преимуществ подхода можно отметить возможность выполнения как процедуру анализа (получение леммы), так и синтеза (генерация нужной словоформы).

Однако у представленных морфологий есть и ряд недостатков:

- Необходимо составление полных и качественных словарей (особенно для языков с высокой флективностью), что требует большого количества человеческих ресурсов.
- Хранение словарей требует значительных объемов памяти при выборе недостаточно эффективной структуры данных.
- Постоянное развитие языка вследствие аккультурации (взаимовлияния культур) осложняет полное покрытие языка - появляются неологизмы, производные слова.
- Отдельную проблему представляет снятие **омонимии** - выбор верного варианта морфологического разбора для токенов, сходных по написанию и звучанию, но имеющих разный смысл, устанавливающий с учетом контекста.

Несмотря на это словарная морфология имеет хорошие показатели и задействована в большинстве практических инструментах.

4.2.3 Конечные преобразователи (FST)

Одним из подходов к морфологическому анализу является применение конечных автоматов и преобразователей. Эта методика основана на правилах и алгоритмах. Существует множество различных технологий и инструментов, использующих конечные

автоматы для решения задач обработки естественного языка, в том числе морфологической анализа.

rule-based методики могут быть более гибкими и адаптивными, чем технологии, основанные на словарях, поскольку они могут работать с неизвестными словами или формами слова, которых нет в словаре. Однако разработка корректного преобразователя, который может обрабатывать все исключительные случаи, очень трудоемкое занятие и возможно не для всех языков.

Формальные понятия

Введем несколько необходимых определений и теорем. В теории формальных языков **регулярный язык (или регулярное множество)** - это множество слов, которые может распознать конечный автомат.

Определение 4.1. Регулярными языками в конечном алфавите $\sum$ называются языки, определяемые по индукции следующим образом:

- Пустое множество $\emptyset$ - регулярный язык
- Множество из пустой строки (ε) - регулярный
- Однобуквенное слово (a), где $a \in \sum$ - регулярный язык
- Пусть α и β - регулярные, тогда следующие операции замкнуты относительно регулярности: конкатенация $\alpha\beta$, объединение $\alpha \cup \beta$, звезда Клини α^*
- Других регулярных языков нет
  - Звезда Клини (замыкание Клини) - операция над множеством V, задаваемая правилом:
    $$V^* = \bigcup_{i=0}^{\inf} V^i$$
    где V^i - конкатенация множества с самим собой i раз.
  - $\alpha^+ = \alpha\alpha^*$ положительное замыкание Клини

Регулярные языки задаются **регулярными выражениями**.

Пример. Пусть $\sum = C, V, \overline{V}$ - (согласный, безударный, ударный). Тогда язык, принимающий слоги, в котором только одна гласная будет задаваться $C^*(V \cup \overline{V})C^*$

Определение 4.2. Конечным автоматом (Finite-state machine FSM) над конечным алфавитом $\sum$ будем называть кортеж $M = \langle Q, \sum, \Delta, q_0 F \rangle$, где

- Q - конечное множество состояний
- $\Delta \subseteq Q \times (\sum \cup \varepsilon) \times Q$ - конечное множество переходов
- $q_0 \in Q$ - стартовое состояние
- $F \subseteq Q$ - завершающие состояния

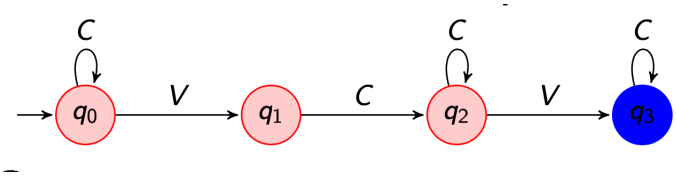

Рис. 2: Автомат для слова с 2 гласными, разделенными хотя бы одним согласным

Определение 4.3. Автомат с однобуквенными переходами - детерминированный, если ни из какого состояние не выходит двух ребер, помеченных одинаковыми буквами

Каждый автоматный язык распознается детерменированным автоматом.

Теорема 4.1. ***Клини*** *Классы автоматных и регулярных языков совпадают*

Теперь перейдем к **конечным преобразователям**. Неформально, это автомат, с добавленными на ребра выходными символами, который можем заменять исходные символы слова на требуемые.

Определение 4.4. Конечный преобразователь (Finite-state transducer FST) над конечными алфавитами $\sum, \Gamma$ - кортеж $T = \langle Q, \sum, \Gamma, \Delta, q_0, F \rangle$, где

- Q - конечное множество состояний
- $\Delta \subseteq Q \times (\sum \cup \varepsilon) \times (\Gamma \cup \varepsilon) \times Q$ - конечное множество переходов
- $q_0 \in Q$ - стартовое состояние
- $F \subseteq Q$ - завершающие состояния

Конечный автомат может быть рассмотрен в как преобразователь, который возвращает входные значения и определяет, принадлежит ли данное входное значение нужному множеству.

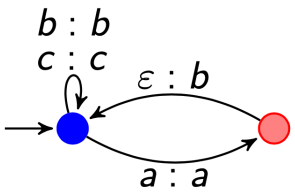

Рис. 3: Преобразователь, который после каждой a добавляет b

Конечные преобразователи замкнуты относительно:

- Композиции: последовательное применение преобразований
- Объединения
- Конкатенации
- Приоритетного объединения $\bigcup_p$:

$$(T_1 \bigcup_p T_2)(u) = \begin{cases} T_1(u), & T_1(u) \neq \emptyset \\ T_2(u), & T_2(u) = \emptyset \end{cases} \tag{1}$$

  Используется для отдельной обработки нерегулярных форм

- Обращения $\varphi^{-1} = \langle y, x \rangle | \langle x, y \rangle \in \varphi$. Используется для перехода от синтеза к анализу и наоборот.

Приложения теории автоматов к морфологическому анализу

Построение конечного автомата для анализа естественного языка - это сложная задача, поскольку естественный язык может иметь множество тонкостей и исключений. В то же время, для агглютинативных языков задача упрощается по нескольким причинам:

- Количество морфем с единственным значением в агглютинативном языке относительно небольшое, что упрощает создание шаблонов для анализа.
- Морфемы в агглютинативных языках выполняют строго определенные функции, поэтому алгоритмы могут быть спроектированы для анализа их комбинаций. Например, в татарском слове "хатларында"морфема "лад"всегда означает множественное число, морфема "ын третьего лица мужского рода, и морфема "да"указывает на местный падеж.

- Паттерны морфем в агглютинативных языках часто имеют небольшую длину, что делает их подходящими для поиска методами конечных автоматов, которые работают с конечным набором символов.

Однако, и для агглютинативных языков эта задача остается непростой. Проблемой является то, что даже в агглютинативных языках могут быть омонимы (слова, которые звучат одинаково, но имеют разные значения), что затрудняет их анализ. Также проблемой является несоответствие между формой слова и его лексическим значением, что порождает исключения и необычные конструкции. К агглютинативным языкам относят: японский, татарский, киргизский, турецкий, эсперанто. А из флексивных выделим: русский, немецкий, польский, испанский, персидский.

Существуют научные исследования по анализу турецкого языка, в которых были созданы морфологические анализаторы TRmorph !!! и TRMOR. Они основаны на подходе двухуровневой морфологии и реализованы с помощью SFST. Авторы !!! описали анализатор, состоит из конечного автомата, правил фонологии и орфографии, а также словаря из 37101 слов, принадлежащих разным частям речи.

Авторы Emmanuel Roche и Yves Schabes в своей статье[3] рассказывают про использование конечных преобразователей (finite transducers) для решения задач автоматического определения грамматической структуры предложений и построения машинного перевода.

В первом случае конечный преобразователь используется для приведения предложения к более удобной форме для анализа его грамматической структуры. Для этого создаются различные конечные преобразователи, каждый из которых выполняет определенную операцию над предложением, например, удаление вспомогательных слов или замена существительных в именительном падеже на иной. В результате применения нескольких таких преобразований получается предложение, которое легче проанализировать.

Для решения задачи машинного перевода используется для перевода предложения из одного языка на другой. Для этого создается пара конечных преобразователей - один для перевода с исходного языка на промежуточный, и другой для перевода с промежуточного языка на целевой язык. Таким образом, перевод осуществляется в два этапа.

Описанный подход позволяет достичь хороших результатов в задачах обработки естественного языка, при этом используя относительно простую модель - конечный пре-

образователь. Кроме того, создание таких инструментов не требует большого объема размеченных данных, что является еще одним преимуществом использования математических моделей для оптимизации анализа.

А в исследовании "Finite-State Transducers in Language and Speech Processing"[4] Mehryar Mohri рассказывает про применения FST:

1. Компиляция исходного текста в FST: исходный текст может быть скомпилирован в конечный преобразователь, используя один из алгоритмов компиляции, описанных в статье. Например, строится автомат соответствующий каждому префиксу строки, которая должна быть распознана, а затем эти автоматы объединяются в единый.

2. После того, как исходный текст был скомпилирован в FST, можно использовать автомат для анализа входного текста. Алгоритм поиска наилучшего пути (Best-Path Search) позволяет найти наилучший путь через FST, соответствующий входному тексту. Это позволяет осуществлять распознавание и классификацию входных строк.

3. И конечные преобразователи используются непосредственно для генерации новых строк на основе входного текста. Алгоритм для генерации всех возможных путей (All-Paths Generation) позволяет сгенерировать все возможные выходные строки, соответствующие входному тексту.

4.2.4 Статистический анализ

!!!!!ДОПИСАТЬ ПРО СТАТИСТИЧЕСКИЕ МЕТОДЫ

4.3 Обзор инструментов морфологического анализа

Необходимость обработки большого объёма текстовой информации приводит к постоянному развитию инструментов автоматического анализа текста. Для эффективной обработки текста, поступающего от пользователей или имеющегося в системах, необходимо улучшать точность и скорость работы лингвистических инструментов, а также использование памяти. В основе инструментов морфологического анализа лежат на словарные и вероятностные подходы, конечные автоматы, машинное обучение и т.д.. Разнообразие схем объясняется тем, что в рамках морфологического анализа решаются различные задачи.

В данном разделе мы рассмотрим несколько таких программных решений для морфологического анализа, которые имеют свою реализацию на Python. Рассмотрение методов, используемых в данных библиотеках, поможет лучше понять особенности морфологического анализа текста и выбрать оптимальное решение для конкретной задачи.

4.3.1 Библиотека pymorphy2

pymorphy2[5] - это библиотека с открытым исходным кодом для морфологического анализа текстов на естественном языке. Она разработана на языке Python и позволяет выполнять различные операции с русским языком, такие как определение частей речи, склонение и спряжение слов, а также получение базовой формы слова (леммы). Библиотека поддерживает русский и украинский языки и является одним из самых популярных инструментов для работы в рамках сообщества Python.

Начало работы и функционал

Установим библиотеку с помощью команды pip. Словари распространяются отдельными пакетами и требуют периодических обновлений, т.к. они могут дополняться.

```
pip install pymorphy2

pip install -U pymorphy2-dicts-ru
pip install -U pymorphy2-dicts-uk
```

Listing 1: Установка и обновление словарей

Для морфологического анализа используется объект класса `MorphAnalyzer` и его метод `.parse()`

```
import pymorphy2
morph = pymorphy2.MorphAnalyzer()
morph.parse('мыла')
```

Разберем структуру ответа на примере выполнения программы выше

```
[Parse(word='мыла', tag=OpencorporaTag('NOUN,inan,neut sing,gent'),
 normal_form='мыло', score=0.333333, methods_stack=((DictionaryAnalyzer(), 'мыла', 54, 1),)),
 Parse(word='мыла', tag=OpencorporaTag('VERB,impf,tran femn,sing,past,indc'),
 normal_form='мыть', score=0.333333, methods_stack=((DictionaryAnalyzer(), 'мыла', 2074, 8),)),
 Parse(word='мыла', tag=OpencorporaTag('NOUN,inan,neut plur,nomn'),
 normal_form='мыло', score=0.166666, methods_stack=((DictionaryAnalyzer(), 'мыла', 54, 6),)),
 Parse(word='мыла', tag=OpencorporaTag('NOUN,inan,neut plur,accs'),
 normal_form='мыло', score=0.166666, methods_stack=((DictionaryAnalyzer(), 'мыла', 54, 9),))]
```

Listing 2: Результат выполнения программы

Структура ответа состоит из поля:

- `word` – исходное слово;
- `tag` – грамматические характеристики;
  - например `OpencorporaTag('VERB,perf,intr plur,past,indc')` дает следующую информацию: слово - глагол (VERB) совершенного вида (perf), непереходный (intr), множественного числа (plur), прошедшего времени (past), изъявительного наклонения (indc). **СДЕЛАТЬ ТАБЛИЦУ С ОБОЗНАЧЕНИЕМ ДЛЯ ГРАММЕМ!!!!!**
- `normal_form` – начальная форма слова;
- `score` - это оценка $P(tag|word)$ вероятности того, что данный разбор правильный.
- `methods_stack` - алгоритм разбора
  - в нашем случае слово было найдено в словаре OpenCorpora, т.е. метод разбора `DictionaryAnalyzer()`

Выбор правильного разбора слова

Анализатор нам выдал несколько вариантов разбора слова 'мыла'. Это происходит из-за невозможности снятия омонимии без контекста. Для выбора наиболее вероятного разбора мы ориентируемся на параметр `score`.

Подсчет условной вероятности $P(tag|word)$ происходит по корпусу `OpenCorpora`[6]: смотрим на частоту слова со снятой омонимией с определенным тегом относительно общей частоты использования слова в корпусе. Добавляется сглаживание Лапласа, чтобы избежать нулевых вероятностей разбора.

$$P(tag|word) = \frac{N(tag, word) + 1}{N(word) + R(word)}$$

где $N(tag, word)$ - количество раз, когда данная словоформа $word$ встретилась с тегом (т.е. с данными грамматическими характеристиками) tag в корпусе, $N(word)$ - количество раз, когда просто встретилась данная словоформа (уже без учета тега), $R(word)$ - число выведенных разборов анализатора для нашего слова $word$ (для корректного определения вероятности).

Хотя оценки вероятностей $P(tag|word)$ полезны для улучшения разбора, но их недостаточно для достоверного определения значения слова в контексте, поскольку:

- правильный вывод зависит от контекста, а не только от самого слова, а библиотека `pymorphy2` работает только с отдельными словами;
- вероятности $P(tag|word)$ основаны на анализе сбалансированных текстовых корпусов, но в специализированных текстах эти вероятности могут отличаться;
  - например, в металлургических текстах вероятность того, что "стали существительное, скорее всего выше, чем вероятность того, что это глагол;
- неоднозначность все еще не снята у большинства слов;

Обработка и хранение словарей

Все словоформы для определенного стема в `OpenCorpora` объединены в лексемы, где указана словоформа и ее грамматическая информация.

```
  хомяковый   ADJF,Qual masc,sing,nomn
  хомякового  ADJF,Qual masc,sing,gent
              ...
  хомяковы    ADJS,Qual plur
  хомяковее   COMP,Qual
  хомяковей   COMP,Qual V-ej
похомяковее   COMP,Qual Cmp2
похомяковей   COMP,Qual Cmp2,V-ej
```

Listing 3: Пример лексемы в словаре OpenCorpora

Если просто выполнять поиск по данным лексем без обработки, то программа будет затрачивать очень много ресурсов по памяти (около 2Гб оперативной памяти) и времени (в словаре около 400 тысяч лексем и 5 миллионов слов). Рассмотрим, какие методы используются для оптимизации работы обработчика[7].

Парадигмы

В рамках одной лексемы можно сделать следующее разбиение у всех словоформ: $word_i =< prefix_i > + < stem > + < suffix_i >$, где $prefix_i$ и $suffix_i$ могут изменяться, за счет чего мы получаем разные словоформы, а $stem$ является общей для всех слов частью (при этом она не обязана совпадать с корнем). $stem$ можно откинуть, и тогда мы получим образец для склонения слов с такими же грамматическими признаками - *парадигму*. Для примера 3 получим:

```
prefix          suffix   tag
                 ый      ADJF,Qual masc,sing,nomn
                 ого     ADJF,Qual masc,sing,gent
                 ...
                 ы       ADJS,Qual plur
                 ее      COMP,Qual
                 ей      COMP,Qual V-ej
    по           ее      COMP,Qual Cmp2
    по           ей      COMP,Qual Cmp2,V-ej
```

Listing 4: Парадигма для лексемы

Заметим, что начальная форма слова располагается в первой строке.

Отдельно в списках хранятся строки, содержащие префиксы, суффиксы и теги, и каждому элементу из шаблона присваивается ее индекс в этом списке - таким образом мы можем хранить парадигму как массив чисел, сначала перечисляя все номера суффиксов, далее тегов и в конце - префиксов.

DAFSA

РАСПИСАТЬ ТЕОРИЮ ПРО DAFSA!!!!!

Для хранения информации о словах используется конечный автомат, также известный как Deterministic Acyclic Finite State Automaton[8]. Для работы с этой структурой данных используется библиотека DAWG-Python. DAWG имеет ряд преимуществ перед другими структурами данных, такими как хэш-таблицы или бинарные деревья поиска. В частности, DAWG позволяет быстро находить все слова, начинающиеся с заданного префикса, а также выполнять операции объединения и пересечения множеств слов. Кроме того, DAWG можно эффективно компактно хранить на диске, что позволяет загружать его в память по требованию. Использование DAWG в Pymorphy2 позволяет

достигать высокой скорости обработки текста и экономить память при работе с большими объемами данных. Автомат строится для строк вида:

< слово >< разделитель >< номер парадигмы >< номер формы в парадигме >

Примеры таких строк:

```
двор    (103, 0)
ёж      (104, 0)
дворник (101, 2) и (102, 2)
ёжик    (101, 2) и (102, 2)
```

Построим для них граф DAWG

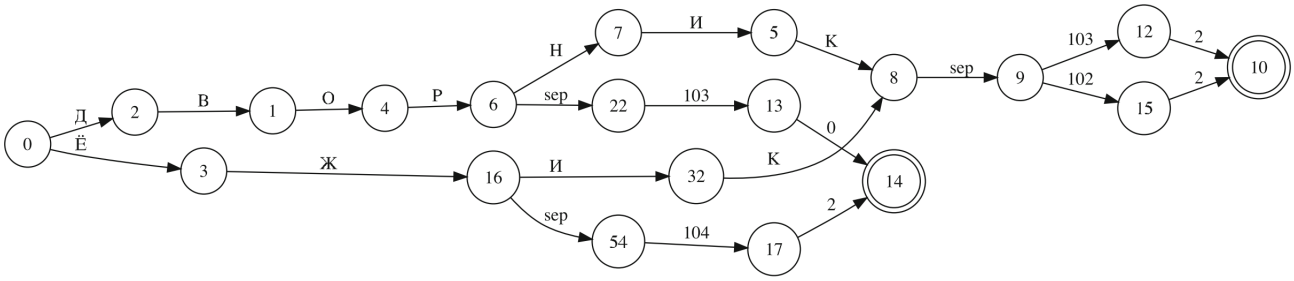

Рис. 4: DAWG-граф для словоформ <двор (103, 0)>, <ёж (104, 0)>, <дворник (101, 2) | (102, 2)>, <ёжик (101, 2) | (102, 2)>

Тогда **алгоритм разбора словарных слов** становится очень простым:

1. Проходом по графу по ключу <СЛОВО><sep> получаем список кортежей $[(para_id1, index$
2. По имеющимся данным о номере парадигмы и номере формы в парадигме получаем начальную форму и грамматические характеристики

Разбор несловарных слов

Для анализа объектов, которые не были найдены алгоритмом поиска по словарю, используется ряд правил, применяющихся последовательно:

1. **Отсечение известных префиксов**. `pymorphy2` хранит список этих словообразовательных префиксо, которые не меняют грамматическое значение слова, и отсекает их при анализе слова, чтобы разобрать остаток и затем вернуть префикс обратно.

2. **Отсечение неизвестных префиксов**. Если первый метод не дал результата, то следующая попытка заключается в выполнении аналогичных действий для префикса длиной от 1 до 5 символов. Этот префикс может быть неизвестным, но все равно будет обработан аналогичным образом. При этом должно учитываться дополнительное условие, что длина словарного слова не должна быть меньше 3 и оно должно быть прилагательным, существительным, глаголом, причастием или деепричастием.

3. **Предсказание по концу слова**. Подходы с отсечением префиксов имеют два ограничения: разбор не должен зависеть от префикса, а морфологический анализатор должен уметь разбирать правую часть слова. Для предсказания форм слов по окончанию, `pymorphy2` использует статистику по окончаниям в словаре и сохраняет только самый частотный разбор для каждой части речи. Разбор сводится к поиску наиболее длинной правой части разбираемого слова в DAFSA с окончаниями. Кроме того, для каждого словоизменительного префикса (ПО, НАИ) строится отдельный DAFSA, чтобы добавить варианты предсказания для таких слов.

МОЖНО ДОПИСАТЬ ПРО DAFSA С ОКОНЧАНИЯМИ !!!!!

При прогнозировании слов на основе их окончания, результаты анализа сортируются по степени "продуктивности"вариантов разбора. Это означает, что более продуктивные варианты появятся в начале списка. То есть, номера парадигм упорядочиваются по частоте их соответствия данному окончанию для данной части речи. При этом не учитывается общая частотность номеров парадигм в корпусе текстов. Несмотря на то, что данный подход не имеет экспериментального подтверждения, он основан на логике: предсказатель должен правильно определять редкие слова, а не предполагать, что они распространены в языке. В то же время, продуктивность парадигм важна, потому что некоторые из них порождают больше слов, чем другие.

НАПИСАТЬ ПОДВОДКУ !!!!!! НАПИСАТЬ ПРО НАРЕЧИЯ и ИНИЦИАЛЫ

В конечном итоге мы получаем хороший инструмент для анализа естественного языка, который быстро работает и может обрабатывать большие объемы данных. Библиотека имеет высокую точность в определении частей речи, склонения и других морфологических признаков слова. `pymorphy2` проста в использовании и предоставляет удобный интерфейс для доступа к морфологической информации. В документации к данному пакету находится подробное описание всех доступных методов с примерами.

А наличие открытого исходного кода позволяет разработчикам широко использовать и улучшать её для своих проектов.

4.3.2 Библиотека pymystem3

В этом модуле представлена обертка для `Yandex Mystem 3.1`[9] - морфологического анализатора русского языка, который базируется на словаре Национального Корпуса Русского Языка (НКРЯ). С помощью данного анализатора можно производить лемматизацию текста и получать набор морфологических характеристик для каждой единицы текста. Кроме того, модуль также поддерживает польский и английский языки. Основными методами в `pymystem3` являются:

- `analyze(text)` - метод, который принимает на вход текст и возвращает список словарей, каждый из которых представляет отдельное слово с его леммой и грамматическими характеристиками.
- `lemmatize(text)` - результатом работы является список лемм (основных форм слов).

Посмотрим примеры использования функционала.

```
from pymystem3 import Mystem

mystem = Mystem()
lemmas = mystem.lemmatize("Мыла раму мою мылом")
print(lemmas)

>>> ['мыть', ' ', 'рама', ' ', 'мой', ' ', 'мыло', '\n']
```

Listing 5: Лемматизация методом `lemmatize()`

Используем также метод для извлечения грамматических признаков и посмотрим на структуру ответа

```
analysis = mystem.analyze("Мыла раму мою мылом")
print(analysis)
```

```
>>>
[{'analysis': [{'lex': 'мыть', 'wt': 0.441520999, 'gr': 'V,несов,пе=прош,ед,изъяв,жен'}],
'text': 'Мыла'}, {'text': ' '},
{'analysis': [{'lex': 'рама', 'wt': 0.9993591156, 'gr': 'S,жен,неод=вин,ед'}], 'text': 'раму'},
{'text': ' '},
{'analysis': [{'lex': 'мой', 'wt': 0.9961201802, 'gr': 'APRO=вин,ед,жен'}], 'text': 'мою'},
{'text': ' '},
{'analysis': [{'lex': 'мыло', 'wt': 1, 'gr': 'S,сред,неод=твор,ед'}], 'text': 'мылом'},
{'text': '\n'}]
```

Listing 6: Грамматический анализ методом `analyze()`

Ключ `text` содержит саму словоформу, которую мы проанализировали. По `analysis` получаем список словарей с информацией о морфологических характеристиках словоформы.

- `lex` - найденная лемма;
- `gr` - строка, где каждый символ обозначает определенную грамматическую характеристику (род, число, падеж, время, наклонение и т.д.). Например "S,сред,неод=им,ед"оз
что это существительное среднего рода, неодушевленное, именительного падежа, единственного числа.
- `wt` - вероятность правильного разбора

Алгоритм разбора

Словарь представлен в виде набора деревьев[10] инвертированных основ ($stem \rightarrow mets$) и дерева всевозможных суффиксов. Граница стема и флексии берется либо из входного файла, если она явно представлена, либо вычисляется автоматически как длина общей части всех словоформ в парадигме.

Погружаемся в слово с правого конца, используя дерево суффиксов. В результате за один проход мы имеем все возможные позиции границы $stem|suffix$. Начнем с самой глубокой возможной границы и повторим шаги:

1. Используем две последние буквы возможной основы в качестве индекса для получения соответствующего дерева перевернутых основ. Если нужная комбинация символов не была найдена, то переходим к следующей границе раздела.
2. В случае успеха предыдущего пункта проходим до конца слова по дереву

- Если мы в попадаем в терминальное состояние, то мы смотрим на идентификатор (хранится как специальная буква после первой буквы стема), который дает информацию о модели суффикса. В случае совпадения ее с обходом по дереву флексий, считаем, что мы нашли **словарное слово**.
- При отсутствии терминального состояния или несовпадении модели с суффиксом: нужно пройти по всем точкам ветвления дерева основ и собрать возможные флексивные модели; сравнить их с искомой.

Интересным применением этого алгоритма является комбинированный словарь нескольких языков, который полезен в ситуациях, когда язык неоднозначен, поэтому комбинированный словарь будет выбирать наиболее вероятные морфологические толкования на основе вероятности для всех слов всех языков. Кроме того, можно добавить собственные словари, которые полностью заменят стандартный словарь.

Снятие омонимии

Морфологический анализатор снимает омонимии в зависимости от учета контекста двумя способами.

Используя размеченный корпус со снятой омонимией, наивный байесовский классификатор обучается для решения задачи снятия омонимии, *не учитывая контекст*. Вспомним, что $word =< stem > + < flex >$, и слову или флексии соответствует определенная $paradigm$ словоизменения. Тогда вероятность слова $word$ принадлежать парадигме $paradigm$ будет рассчитываться следующим образом:

$$P(paradigm|word) = \frac{P(word|paradigm) \cdot P(paradigm)}{P(word)} =$$

$$= \frac{P(stem|paradigm) \cdot P(flex|paradigm) \cdot P(paradigm)}{P(word)}$$

В Mystem *контекстное снятие* омонимии осуществляется через технологию MatrixNet. Данная технология используется в многих продуктах компании и является одной из самых эффективных методов снятия омонимии. В алгоритме также используются дополнительные эвристики, включающие:

- обязательное наличие гласной в полученной основе;
- установление минимальной длины основы и минимальной общей части между примером и неизвестным словом;

- учитывание наличия парадигмы модели в словаре не менее двух раз.

5 Описание эксперимента

Для проведения эксперимента были выбраны два морфологических анализатора `pymorphy2` и `pymystem3`, методы и устройство которых были описаны в предыдущей главе 4. В работе будет рассмотрено сравнение скорости работы, точности определения частей речи и граммем. Результаты исследования помогут определить наиболее подходящий инструмент для решения конкретных задач.

5.0.1 Тестовые данные

Для проверки работоспособности инструментов необходимы специальные наборы тестовых данных с морфологической разметкой. В рамках этого исследования были рассмотрены уже существующие корпуса текстов на русском языке, которые могут быть использованы в качестве таких тестовых данных.

1. National Corpus of Russian (Национальный Корпус Русского Языка). Самый крупный корпус текстов русского языка с морфологической разметкой. Собрал в себе более 220 млн словоупотреблений на разных языковых уровнях, включая лексические, грамматические и синтаксические данные. Содержит тексты на различные тематики, в том числе научные, художественные, публицистические и др. Объем текстов с морфологической разметкой составляет около 70 млн словоупотреблений. При этом подкорпус со снятой омонимией содержит около 1 млн словоупотреблений.

2. OpenCorpora. Проект базируется на открытом доступе к большому количеству текстов для создания общего и доступного ресурса морфологической разметки. Он содержит более трех миллионов словоформ и более 400 тысяч лемм (словоформ, приведенных к начальной форме). Тексты включают в себя различные тематики: научные, художественные, деловые и др. Объем текстов со снятой омонимией составляет около 30%.

3. СинТагРус. Это корпус текстов на русском языке, построенный на базе деревьев составленных на словах, где каждое слово помечено синтаксической информацией, морфологическим тегом и начальной формой слова. Содержит около 150 тысяч предложений, объем текстов со снятой омонимией составляет около 85%.

Выбор тестовых данных пал на OpenCorpora, не будем лукавить, в первую очередь из-за его доступности. Однако невозможно не выделить плюсы данного выбора: корпус

содержит разнообразные данные по типам текстов, имеет высокое качество разметки и понятный для парсинга интерфейс, а так же он достаточно репрезентативен - это компиляция текстов, написанных реальными людьми в разных контекстах.

5.0.2 Предобработка данных

Для того, чтобы получить точную оценку качества работы морфологического анализатора, необходимо предварительно выполнить предобработку морфологически размеченного текста:

1. Удалить все знаки препинания, поскольку они не влияют на морфологический анализ.

2. Исключить все слова, которые не являются словоформами, т.е. слова с тегами, отличными от частей речи (например, числительные, местоимения и т.д.).

3. Исключить все словоформы с некорректными или неопределенными тегами, чтобы избежать ошибок при последующей оценке.

4. Привести все словоформы к единому регистру (например, к нижнему), чтобы исключить несоответствия регистра при сравнении.

5. Объединить все словоформы в предложения, сохраняя порядок слов, чтобы сохранить контекст и соответствие между словами.

6. Удалить из предложений все стоп-слова, которые не несут особого значения для морфологического анализа (например, союзы, предлоги).

Вся предобработка выполняется исходя из структуры поставляемых размеченных текстов. А именно: опираемся на теги OpenCorpora для исключения всех требуемых слов и символов.

5.0.3 Внутреннее устройство разметки

Тексты организованы в иерархическом порядке, где короткие статьи или новостные заметки составляют единый текст, а более длинные произведения могут быть разбиты на несколько текстов в зависимости от соответствующей структуры (например, главы или разделы).

Каждый текст может иметь несколько тегов, содержащих информацию о тексте в целом, такую как автор, дата создания, источник (при наличии URL) и другие. Тексты

разделены на абзацы, которые определены при источнике, и на предложения, которые были разделены вручную.

Формат разметки в OpenCorpora представлен в виде XML-документа, который содержит данные о каждом слове в тексте. В разметке используются различные теги, которые обозначают конкретные части речи или грамматические категории.

```
<sentence id="1">
    <source>«Школа злословия» учит прикусить язык</source>
    <tokens>
      <token id="1" text="«"><tfr rev_id="2420236" t="«"><v><l id="0" t="«"><g v="PNCT"/></l></v></tfr></token>
      <token id="2" text="Школа"><tfr rev_id="834910" t="Школа"><v><l id="380220" t="школа"><g v="NOUN"/>
      <g v="inan"/><g v="femn"/><g v="sing"/><g v="nomn"/></l></v></tfr></token>
      <token id="3" text="злословия"><tfr rev_id="2632816" t="злословия"><v><l id="115766" t="злословие"><g v="NOUN"/>
      <g v="inan"/><g v="neut"/><g v="sing"/><g v="gent"/></l></v></tfr></token>
      <token id="4" text="»"><tfr rev_id="2420237" t="»"><v><l id="0" t="»"><g v="PNCT"/></l></v></tfr></token>
      <token id="5" text="учит"><tfr rev_id="834913" t="учит"><v><l id="363313" t="учу"><g v="VERB"/>
      <g v="impf"/><g v="tran"/><g v="sing"/><g v="3per"/><g v="pres"/><g v="indc"/></l></v></tfr></token>
      <token id="6" text="прикусить"><tfr rev_id="834914" t="прикусить"><v><l id="271426" t="прикусить"><g v="INFN"/>
      <g v="perf"/><g v="tran"/></l></v></tfr></token>
      <token id="7" text="язык"><tfr rev_id="3408577" t="язык"><v><l id="387573" t="язык"><g v="NOUN"/>
      <g v="inan"/><g v="masc"/><g v="sing"/><g v="accs"/></l></v></tfr></token>
    </tokens>
</sentence>
```

Listing 7: XML-дерево с разметкой для одного предложения

Для парсинга дерева был выбран модуль `xml.etree.ElementTree`. Для получения тегов пользуемся методом `.findall()`. Поиск осуществляется по пути

path = 'text/paragraphs/paragraph/sentence/tokens/token/tfr/v/l/g'

5.0.4 Отображение граммем

При обзоре различных программ-анализаторов возникает проблема унификации грамматических единиц. В русском языке определенный тип слов (предикатив) может функционировать как наречие и другая часть речи. Из-за этого нельзя однозначно определить часть речи каждого слова. Обычно наречие функционирует в качестве предикатива, таким образом, оно использовалось для отображения. Для того чтобы решить эту проблему, все инструменты были проанализированы и отображены на определенное множество, которое используется в корпусе. Таблица с отображением граммем представлена ниже 1.

Таблица 1: Отображение граммем и частей речи

Opencorpora	**Mystem**	**pymorphy2**
NOUN	S	NOUN
VERB, INF, PRTF, PRTS, GRND	V	VERB, INF, PRTF, PRTS, GRND
NUMR	NUM	NUMR, NUMB, ROMN
ADJF, ADJS, COMP	A, ANUM, APRO	ADJF, ADJS, COMP
ADVB, PRED	ADV, ADVPRO	ADVB, PRED
PREP	PR	PREP
CONJ	CONJ	CONJ
PRCL	PART	PRCL
INTJ	INTJ	INTJ
NPRO	SPRO	NPRO

5.0.5 Результаты

Эксперимент был проведен в несколько этапов. На первом этапе была удалена пунктуация и неопределенные теги, а так же символы и латиница. Далее были удалены числительные, местоимения, союзы и предлоги. В последнюю очередь были исключены все токены, относящиеся к нумерации и времени. Результаты для метрики *accuracy* сведены в таблицу: 2.

Таблица 2: *accuracy* для набора экспериментов для pymorpy2 и pymystem3

Анализатор	**Эксперимент 1**	**Эксперимент 2**	**Эксперимент 3**
pymorphy2	0.926	0.913	0.978
pymystem3	0.833	0.892	0.955
Число слов после обработки	64736	50767	47291

6 Заключение

Mystem - Данный морфологический анализатор расчитан на максимальное быстродействие для использования в поисковом движке. Поэтому о каких-то серьёзных использованиях контекста речи не идет.

Segalovich I. A fast morphological algorithm with unknown word guessing induced by a dictionary for a web search engine //MLMTA. – 2003. – С. 273-280.

Заключение. В рамках данной статьи был сделан обзор трех морфологических анализаторов, даны их программные характеристики, обозначена актуальность их применения в процессе автоматической обработки текстов. Для рассмотрения был взяты отечественные и иностранный программный продукт, способный работать не только с русским языком. Каждый из них предоставляет основной набор грамматической и лингвистической информации о слове, однако есть различия в дополнительных параметрах.

Список литературы

[1] *Chomsky, Noam.* Morphophonemics of modern Hebrew / Noam Chomsky. — Taylor and Francis, 2013. — 01. — Publisher Copyright: © 1979 Avram Noam Chomsky. All rights reserved.

[2] *Hutchins, W. John.* The Georgetown-IBM Experiment Demonstrated in January 1954 / W. John Hutchins // Machine Translation: From Real Users to Research / Ed. by Robert E. Frederking, Kathryn B. Taylor. — Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. — Pp. 102–114.

[3] *Roche, Emmanuel.* Deterministic Part-Of-Speech Tagging With Finite State Transducers / Emmanuel Roche, Yves Schabes // *Computational Linguistics.* — 1995. — Vol. 21, no. 2. — Pp. 227–253. https://aclanthology.org/J95-2004.

[4] *Mohri, Mehryar.* Finite-State Transducers in Language and Speech Processing / Mehryar Mohri // *Computational Linguistics.* — 1997. — Vol. 23, no. 2. — Pp. 269–311. https://aclanthology.org/J97-2003.

[5] pymorphy2 [Электронный ресурс]. — https://pymorphy2.readthedocs.io/en/stable/.

[6] Открытый корпус OpenCorpora [Электронный ресурс]. — http://www.opencorpora.org/.

[7] *Korobov, Mikhail.* Morphological Analyzer and Generator for Russian and Ukrainian Languages / Mikhail Korobov // Analysis of Images, Social Networks and Texts / Ed. by Mikhail Yu. Khachay, Natalia Konstantinova, Alexander Panchenko et al. — Springer International Publishing, 2015. — Vol. 542 of *Communications in Computer and Information Science.* — Pp. 320–332. http://dx.doi.org/10.1007/978-3-319-26123-2_31.

[8] Incremental Construction of Minimal Acyclic Finite-State Automata / Jan Daciuk, Stoyan Mihov, Bruce W. Watson, Richard E. Watson // *Computational Linguistics.* — 2000. — 03. — Vol. 26, no. 1. — Pp. 3–16. https://doi.org/10.1162/089120100561601.

[9] Морфологический анализатор Mystem 3.0 [Электронный ресурс]. — https://yandex.ru/dev/mystem/.

[10] Introduction to Algorithms / Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. — 2 edition. — The MIT Press, 2001.

[11] Национальный корпус русского языка [Электронный ресурс]. — https://ruscorpora.ru/.

[12] *Le, Duc C.* Evaluating insider threat detection workflow using supervised and unsupervised learning / Duc C Le, A Nur Zincir-Heywood // 2018 IEEE Security and Privacy Workshops (SPW) / IEEE. — 2018. — С. 270–275.

[13] Role-based log analysis applying deep learning for insider threat detection / Dongxue Zhang, Yang Zheng, Yu Wen и др. // Proceedings of the 1st Workshop on Security-Oriented Designs of Computer Architectures and Processors. — 2018. — С. 18–20.

[14] *Le, Duc C.* Exploring anomalous behaviour detection and classification for insider threat identification / Duc C Le, Nur Zincir-Heywood // *International Journal of Network Management.* — 2021. — Т. 31, № 4. — С. e2109.

[15] Deep learning based attribute classification insider threat detection for data security / Fanzhi Meng, Fang Lou, Yunsheng Fu, Zhihong Tian // 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC) / IEEE. — 2018. — С. 576–581.

[16] *Le, Duc C.* Anomaly detection for insider threats using unsupervised ensembles / Duc C Le, Nur Zincir-Heywood // *IEEE Transactions on Network and Service Management.* — 2021. — Т. 18, № 2. — С. 1152–1164.

[17] *Al-Shehari, Taher.* An insider data leakage detection using one-hot encoding, synthetic minority oversampling and machine learning techniques / Taher Al-Shehari, Rakan A Alsowail // *Entropy.* — 2021. — Т. 23, № 10. — С. 1258.

[18] An ensemble approach for detecting anomalous user behaviors / Xiangyu Xi, Tong Zhang, Wei Ye и др. // *International Journal of Software Engineering and Knowledge Engineering.* — 2018. — Т. 28, № 11n12. — С. 1637–1656.

[19] Detecting insider threat from enterprise social and online activity data / Gaurang Gavai, Kumar Sricharan, Dave Gunning и др. // Proceedings of the 7th ACM CCS international workshop on managing insider security threats. — 2015. — С. 13–20.

[20] Deep learning for unsupervised insider threat detection in structured cybersecurity data streams / Aaron Tuor, Samuel Kaplan, Brian Hutchinson и др. // *arXiv preprint arXiv:1710.00811*. — 2017.

[21] *Wirth, Rüdiger*. CRISP-DM: Towards a standard process model for data mining / Rüdiger Wirth, Jochen Hipp // Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining / Manchester. — Т. 1. — 2000. — С. 29–39.

[22] A review of insider threat detection: Classification, machine learning techniques, datasets, open challenges, and recommendations / Mohammed Nasser Al-Mhiqani, Rabiah Ahmad, Z Zainal Abidin и др. // *Applied Sciences*. — 2020. — Т. 10, № 15. — С. 5208.

[23] The insider threat to information systems and the effectiveness of ISO17799 / Marianthi Theoharidou, Spyros Kokolakis, Maria Karyda, Evangelos Kiountouzis // *Computers & Security*. — 2005. — Т. 24, № 6. — С. 472–484.

[24] *Bishop, Matt*. Defining the insider threat / Matt Bishop, Carrie Gates // Proceedings of the 4th annual workshop on Cyber security and information intelligence research: developing strategies to meet the cyber security and information intelligence challenges ahead. — 2008. — С. 1–3.

[25] *Cappelli, Dawn M*. The CERT guide to insider threats: how to prevent, detect, and respond to information technology crimes (Theft, Sabotage, Fraud) / Dawn M Cappelli, Andrew P Moore, Randall F Trzeciak. — Addison-Wesley, 2012.

[26] *Glasser, Joshua*. Bridging the gap: A pragmatic approach to generating insider threat data / Joshua Glasser, Brian Lindauer // 2013 IEEE Security and Privacy Workshops / IEEE. — 2013. — С. 98–104.

[27] A survey of deep learning methods for cyber security / Daniel S Berman, Anna L Buczak, Jeffrey S Chavis, Cherita L Corbett // *Information*. — 2019. — Т. 10, № 4. — С. 122.

[28] Learning similarity measure for natural image retrieval with relevance feedback / Guo-Dong Guo, Anil K Jain, Wei-Ying Ma, Hong-Jiang Zhang // *IEEE Transactions on Neural Networks*. — 2002. — Т. 13, № 4. — С. 811–820.

[29] *Yang, Zhilin.* Revisiting semi-supervised learning with graph embeddings / Zhilin Yang, William Cohen, Ruslan Salakhudinov // International conference on machine learning / PMLR. — 2016. — С. 40–48.

[30] *Liu, Fei Tony.* Isolation forest / Fei Tony Liu, Kai Ming Ting, Zhi-Hua Zhou // 2008 eighth ieee international conference on data mining / IEEE. — 2008. — С. 413–422.

[31] *Boser, Bernhard E.* A training algorithm for optimal margin classifiers / Bernhard E Boser, Isabelle M Guyon, Vladimir N Vapnik // Proceedings of the fifth annual workshop on Computational learning theory. — 1992. — С. 144–152.

[32] LOF: identifying density-based local outliers / Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, Jörg Sander // Proceedings of the 2000 ACM SIGMOD international conference on Management of data. — 2000. — С. 93–104.

[33] Support vector method for novelty detection / Bernhard Schölkopf, Robert C Williamson, Alex Smola и др. // *Advances in neural information processing systems.* — 1999. — Т. 12.

[34] *Matthews, Brian W.* Comparison of the predicted and observed secondary structure of T4 phage lysozyme / Brian W Matthews // *Biochimica et Biophysica Acta (BBA)-Protein Structure.* — 1975. — Т. 405, № 2. — С. 442–451.

[35] *Kim, Wonjik.* Unsupervised learning of image segmentation based on differentiable feature clustering / Wonjik Kim, Asako Kanezaki, Masayuki Tanaka // *IEEE Transactions on Image Processing.* — 2020. — Т. 29. — С. 8055–8068.

[36] *Goldberg, Lewis R.* An alternative"description of personality": the big-five factor structure. / Lewis R Goldberg // *Journal of personality and social psychology.* — 1990. — Т. 59, № 6. — С. 1216.

[37] SMOTE: synthetic minority over-sampling technique / Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, W Philip Kegelmeyer // *Journal of artificial intelligence research.* — 2002. — Т. 16. — С. 321–357.

[38] Opinion Lexicon: Negative. — `https://ptrckprry.com/course/ssd/data/negative-words.txt`. — Дата обращения: 12.03.2022.

[39] Insider Threat Test Dataset. cmu. — `https://kilthub.cmu.edu/articles/dataset/Insider_Threat_Test_Dataset/12841247/1`. — Дата обращения: 18.06.2022.

[40] Google Colaboratory. google inc. — https://colab.research.google.com/. — Дата обращения: 17.06.2022.

[41] Project jupyter homepage. project jupyter. — https://jupyter.org/. — Дата обращения: 12.04.2022.

[42] Google Drive. google inc. — https://drive.google.com/. — Дата обращения: 18.06.2022.

[43] Google Dataset Search. google inc. — https://datasetsearch.research.google.com/. — Дата обращения: 07.11.2021.

[44] Kaggle. google inc. — https://www.kaggle.com/. — Дата обращения: 07.11.2021.

[45] Market Guide for User and Entity Behavior Analytics. gartner inc. — https://www.gartner.com/en/documents/3538217. — Дата обращения: 16.03.2022.

[46] Clearswift Insider Threat Index 2017. Clearswift. — https://www.clearswift.com/resources/press-releases/insider-threat-74-security-incidents-come-extended-enterprise-not-hacking-groups. — Дата обращения: 26.05.2022.

[47] WikiLeaks. wikileaks. — https://wikileaks.org/. — Дата обращения: 11.06.2022.

[48] SciKit-Learn: Outlier detection. — https://scikit-learn.org/stable/modules/outlier_detection.html. — Дата обращения: 10.05.2022.

[49] European Parliament resolution of 21 January 2021 with recommendations to the Commission on the right to disconnect (2019/2181(INL)). Access to European Union law. — https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A52021IP0021. — Дата обращения: 11.05.2022.