

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Политехнический институт
Факультет информационных технологий и электроники

Кафедра
«Информационная безопасность систем и технологий»

Специальность 10.05.03 –

Информационная безопасность автоматизированных систем

Специализация -

Защищенные автоматизированные системы управления

ДИПЛОМНАЯ РАБОТА

на тему

ПРОГРАММНОЕ СРЕДСТВО ДЛЯ ФОРМИРОВАНИЯ БЕЗОПАСНЫХ
СЕМАНТИЧЕСКИХ ПАРОЛЕЙ

Студент

Мартиросян Егор Андреевич

Руководитель

Лупанов М.Ю.

Нормоконтролёр

Егорова Н.А.

Рецензент зам. ген. директора по спец. вопросам АО «ПНИЭИ»

6.06.23

Соловьев С.А.

16.06.23

Работа допущена к защите (протокол заседания кафедры от 16.06.23 № 10)

Заведующий кафедрой

С.Л. Зефирова

Работа защищена с отметкой хорошо (протокол заседания ГЭК от 27.06.23 № 4)

Секретарь ГЭК

А.П. Иванов

Пенза 2023

Содержание

Введение.....	3
1 Современные требования к паролям и существующие решения.....	6
2 Расчёт статистических характеристик паролей	14
3 Разработка схемы формирования парольной фразы	25
4 Разработка алгоритма формирования семантических паролей.....	35
5 Библиотеки python для работы с текстовыми данными.....	44
6 Тестирование программы и разработка руководства пользователя	64
7 Вредные психофизиологические факторы, влияющие на психоэмоциональное состояние специалиста по защите информации.	
Методы и средства физической культуры, снижающие их воздействие	71
Заключение	78
Список использованных источников	79
Приложение А. Обязательное. Блок-схема алгоритма генератора семантических паролей.	82
Приложение Б. Обязательное. Листинг программы генератора семантических паролей	84

Введение

В настоящее время информационные технологии уже проникли практически во все сферы жизнедеятельности человека, вынуждая его ежедневно взаимодействовать сразу с несколькими информационными системами. Это приводит к возникновению множества реальных угроз, связанных с несанкционированным доступом к конфиденциальным данным. Сложившаяся ситуация во многом усугубляется широким распространением злонамеренных и противоправных действий в сфере информационных технологий, а также постоянным целенаправленным совершенствованием средств для их осуществления. Несмотря на повсеместное внедрение средств биометрической аутентификации и разнообразных средств аппаратной защиты, надежный пароль и по сей день остаётся наиболее распространённым, а порой - единственным, инструментом для защиты данных от несанкционированного доступа [1].

Всё это заставляет постоянно ужесточать требования к сложности используемых паролей. В настоящее время наиболее распространённым требованием к минимальной длине пароля является 8 символов. При этом для повышения надёжности к паролям часто предъявляются дополнительные требования, такие как использование букв верхнего и нижнего регистра, включение одной или нескольких цифр и специальных символов, запрет слов, найденных в чёрном списке паролей, содержащих личную информацию пользователя или информацию о компании, запрет паролей, совпадающих с форматом даты, номерами машин, телефонными номерами и т.п.

Иногда системы безопасности сами генерируют пароль для пользователя или позволяют ему выбрать один из ограниченного числа вариантов. Некоторые политики безопасности требуют, чтобы пользователь менял пароль периодически, например каждые 90 или 180 дней. При этом современные требования безопасности, как правило, прямо запрещают где-либо записывать пароль к системе и предписывают хранить его только в своей памяти [1].

На данный момент требования настолько ужесточились, что удовлетворение всем им в полной мере стало просто нереалистичным, и они, так или

иначе, неизбежно нарушаются. Основная проблема заключается в том, что пароли достаточной для обеспечения требуемого уровня безопасности сложности с учётом необходимости использования их большого числа обычному человеку просто невозможно запомнить. В этой связи исследование возможностей эффективного использования семантики осмысленных фраз и словосочетаний естественного языка в целях улучшения запоминаемости сложных, устойчивых к взлому паролей, в настоящее время представляет особенно большой интерес, а тема выпускной квалификационной работы "Безопасность семантических паролей" является особенно актуальной.

Уже сейчас в этом направлении имеются интересные наработки, постоянно появляются новые программные продукты, предназначенные для генерации парольных фраз и основанных на них легко запоминающихся сильных паролей. Но все они имеют свои недостатки и нуждаются в совершенствовании и доработке. К тому же большинство подобных программных продуктов рассчитаны на использование только латинской раскладки и мало подходят для русскоязычных пользователей, так как в этих приложениях никак не учитываются особенности построения предложений русского языка.

Целью настоящей работы является разработка и программная реализация генератора надёжных русскоязычных семантических паролей.

В рамках поставленной цели в работе проведён анализ современных требований, предъявляемых к паролям, и проведён обзор существующих решений в данной области; проанализированы основные статистические характеристики паролей, определяющие стойкость к взлому, а также методы их расчета. В соответствии со всеми предъявляемыми требованиями разработана схема формирования семантического пароля, позволяющая генерировать пароли разной длины и использовать для этого фразы разной структуры.

Кроме этого в работе проведён обзор библиотек python для работы с текстовыми данными, разработан и реализован алгоритм генерации семантического пароля. В заключении проведено исследование функционирования

разработанной программы и даны практические рекомендации по её использованию.

1 Современные требования к паролям и существующие решения

В последнее время требования к выбору пароля, обеспечивающего достаточную стойкость к взлому, сильно ужесточились. С одной стороны, это связано с резким ростом вычислительных мощностей, доступных злоумышленникам. С другой стороны — совершенствованием методов взлома паролей и применением для этих целей специализированных систем искусственного интеллекта. Следует учитывать, что короткие пароли с числом символов 6 и менее, в настоящее время являются уязвимыми независимо от сложности своего содержания, так как довольно легко поддаются взлому методом прямого подбора. Поэтому такие пароли использовать просто недопустимо. Вместе с этим в атаках на системы авторизации или программы для хранения паролей криптоанализ и методы прямого подбора используются все реже [3]. Вместо этого всё чаще используются атаки с использованием интеллектуальных систем на основе специально оптимизированных словарей, позволяющих резко сократить число вариантов перебора. Таким атакам может предшествовать сбор информации и анализ информационной среды жертвы, атаки на её дополнительные учётные данные, смарт-карты, устройства-токены, а также - атаки направленные на взаимодействия жертвы с системой авторизации.

Многочисленные факты убедительно подтверждают крайне ограниченную фантазию среднестатистического пользователя, проявляемую им при выборе пароля. Чаще всего для этих целей используются слова и словосочетания, имеющие прямое отношение к пользователю, либо к самой системе авторизации. Иногда по личной инициативе, но чаще с целью удовлетворения минимальным требованиям системы авторизации, выбираемые пароли усложняются или путём изменения регистра символов, добавления цифр и специальных знаков в конец пароля, с помощью транслитерации и т. п. Однако, все эти манипуляции лишь незначительно повышают реальную криптостойкость составленного пароля, так как могут быть легко учтены в алгоритме взлома при сравнительно небольшом увеличении его трудоёмкости [1].

При самостоятельном выборе пароля, пользователь опирается на собственный словарный запас, который для большинства людей ограничен 20 тыс. словами, а частота использования разных слов сильно различается. Поведение среднестатистического пользователя при выборе пароля во многом предсказуемый процесс, который может быть смоделирован. При наличии специальной статистики и достаточного объёма сведений о пользователе, выбранный им пароль может быть предсказан с весьма высокой вероятностью. Поэтому надёжный пароль не должен ассоциироваться с личностью пользователя и самой системой авторизации. Однако, это требование в полной мере может быть обеспечено только при полном отказе пользователя от самостоятельного выбора пароля, и передаче этой функции системе генерации пароля.

Сгенерированные пароли, обладая высокой надёжностью, имеют один серьёзный недостаток — это очень плохая запоминаемость. Накопленный опыт показывает, что пользователям бывает крайне сложно запомнить заранее заданные хаотичные пароли, равно как и пароли высокого уровня сложности, выбираемые ими по определённому предписанию. Особенно остро эта проблема проявляется при редком использовании пароля, так как в отсутствии постоянного повторения, его становится практически невозможно воспроизвести по памяти. Именно по этой причине, вопреки всем запретам, многие пользователи склонны записывать сложные пароли и даже оставлять эти записи в небезопасных местах, зачастую в открытом виде, сводя на нет все преимущества от использования надёжного криптостойкого пароля [4].

Пароли могут сохраняться и в относительно безопасном месте, например в записной книжке, смартфоне или карманном персональном компьютере. К тому же, они могут дополнительно шифроваться. Однако, все эти меры существенно повышают риск потери доступа к паролю, или его полной утраты. Хотя в этом случае последствия обычно оказываются не столь плачевными, как в случае компрометации, так как обычно в системе безопасности существует возможность сброса утраченного пароля и имеется механизм восстановления доступа к системе.

Следует отметить, что мнения разных специалистов в отношении записывания паролей неоднозначны [5], так Джеспер Йохансон, старший специалист по вопросам стратегии в области безопасности Microsoft, даже предложил отменить существующие запреты записывать пароли. Однако, эта инициатива вызвала критику со стороны других специалистов, так как она, по сути, предлагает смириться с многочисленными небезопасными ситуациями, неизбежно возникающими при хранении паролей на физических носителях.

Большинство же специалистов сходятся во мнении, что единственный путь, позволяющий в полной мере преодолеть все сложившиеся противоречия, заключается в использовании специальных мнемонических паролей.

Мнемоническими принято называть легко запоминающиеся, благодаря применению специальных мнемонических приёмов, пароли. Например, пароль может быть составлен из первых букв легко запоминающегося высказывания, стихотворения, песни, поговорки и т. п. Необходимым условием эффективного использования таких паролей является возможность лёгкого воспроизведения заложенных в их основу фраз по памяти. Неоднократно проводимые исследования в этой области показали, что составленные таким образом пароли, при определённых условиях, по своим свойствам могут быть очень схожи с паролями, составленными из случайно выбранных символов.

Мнемонические пароли не лишены недостатков обычных паролей, в частности, они могут повторно использоваться сразу в нескольких системах авторизации. К тому же, хотя и в меньшей степени, они также подвержены интеллектуальным атакам с использованием оптимизированных словарей. Слабым местом мнемонических паролей является тот факт, что для их составления пользователи предпочитают использовать хорошо известные фразы и отрывки текста. Задача взлома в какой-то степени здесь даже упрощается, так как не требуется восстанавливать саму фразу, порождающую мнемонический пароль. К настоящему моменту уже разработаны разнообразные специализированные словари для подбора таких мнемонических паролей.

И тем не менее, мнемонические пароли имеют существенное преимущество перед обычными паролями, заключающееся в их лёгком запоминании и отсутствии необходимости записывания на физическом носителе. Это преимущество достигается за счёт использования особенностей человеческого мышления, способного гораздо проще запомнить достаточно длинную, но осмысленную фразу, по сравнению с коротким, но бессмысленным набором символов. Помимо простых мнемонических паролей существуют мнемонические пароли повышенного уровня безопасности. Они представляют собой такие же пароли, образованные от простых и легко запоминающихся слов и словосочетаний, но в них дополнительно используются специальные мнемонические замены, преследующие целью сделать пароль как можно более сложным и устойчивым к взлому. Одним из примеров этого метода является замена латинских букв в формирующих пароль словах внешне похожими на них цифрами или специальными символами, такими как 3, @, 0, & и т. д.

Однако, далеко не все пароли могут быть преобразованы таким образом, что накладывает существенные ограничения либо на объём используемого словаря, либо на сложность формируемого пароля. К тому же, мнемонические пароли повышенного уровня безопасности основаны на преобразованиях легко запоминающихся слов из специальных словарей. Злоумышленники уже давно разработали специальные словари для подбора паролей, основанных на всевозможных преобразованиях. При этом, как и обычные мнемонические пароли, пароли с повышенным уровнем безопасности могут использоваться сразу в нескольких системах авторизации. Общая тенденция свидетельствует о том, что чем качественнее и надежнее пароль, тем большую ценность он представляет для пользователя, тем менее охотно он с ним расстаётся, и более склонен к повторному использованию этого пароля.

В последнее время очень широкое распространение получили фразовые пароли. Именно они представляются наиболее перспективным и предпочтительным вариантом мнемонических паролей, так как в сравнении с обычными паролями пользователю их гораздо проще запомнить [3].

К тому же такие пароли могут быть гораздо длиннее, и это существенно повышает их стойкость к атакам прямого перебора. Фразовые пароли в целом более сложны, так как в них могут использоваться знаки пунктуации, пробелы, а также символы в разных регистрах. Вместе с этим, использование парольных фраз имеет свои ограничения. Так, далеко не все системы безопасности поддерживают возможность введения очень длинных паролей, а сам процесс введения таких паролей неудобен и сопровождается повышенным риском ошибок ввода. К тому же, стойкость длинной парольной фразы, определяется не количеством входящих в её состав символов, как это может показаться на первый взгляд, а лишь количеством слов, входящих в её состав, и мало чем отличается от криптостойкости мнемонических паролей.

Иногда для улучшения запоминаемости и достижения высокой криптостойкости пароля применяются мнемонические формулы [6], представляющие собой особую технику использования заданной, легко воспроизводимой по памяти схемы для оперативного создания пароля, основанного на доступных конкретному пользователю данных. При использовании хорошей мнемонической формулы формируемый пароль мало чем отличается от случайного набора символов, а его сложность и длина оказываются вполне достаточными для того, чтобы обеспечить гарантированную устойчивость ко всем разновидностям взлома паролей. Вместе с этим, мнемоническая формула обеспечивает простое восстановление пароля по памяти при известной целевой системе авторизации и наличии специфических данных авторизуемого пользователя. Формируемый таким образом пароль может быть уникален для каждого пользователя, каждого уровня доступа и каждой системы авторизации, что заметно повышает его надёжность и универсальность. В [6] приводится множество вариантов разнообразных мнемонических формул.

Идея использования мнемонических формул довольно популярна и часто рекомендуется для преодоления плохой запоминаемости пароля [7], однако очевидно, что это решение не отвечает базовым положениям информационной безопасности, и следовательно не может рассматриваться всерьёз.

Во-первых, при использовании мнемонической формулы, как правило прямо рекомендуется использовать информацию, связанную с авторизуемым пользователем и системой авторизации, чего по соображениям информационной безопасности, наоборот, следует избегать. Но более серьезным недостатком является то, что стойкость пароля к взлому, по большей части, определяется самой мнемонической формулой, по которой он был сгенерирован.

Автор статьи [6] прямо пишет, что раскрытие мнемонической формулы практически равносильно раскрытию сгенерированного с его помощью пароля, так как остальная информация, используемая для его генерации находится в открытом доступе. При анализе криптостойкости систему шифрования принято считать известной злоумышленнику, а её сокрытие рассматривается лишь как дополнительная мера, повышающая надёжность системы безопасности. Это связано с тем, что систему шифрования невозможно долго скрывать. Особенно при регулярном использовании множеством пользователей. Мнемоническую формулу следует рассматривать именно как часть системы шифрования, а не часть пароля, так как в [6] и других источниках предполагается возлагать функции её формирования и распространения на администрацию информационной системы, что по сути открывает ей доступ к паролям, и полностью дискредитирует всю систему безопасности. По общему правилу безопасности, пароль должен быть известен только пользователю, остальные же субъекты, независимо от статуса, должны находиться в равном положении относительно возможности получения доступа к паролю.

Таким образом, мнемонические формулы, если и могут сколь либо эффективно применяться, то только для индивидуального использования. При этом они должны рассматриваться как составная часть пароля, а не способы его генерации. В этом случае каждая смена пароля должна обязательно сопровождаться сменой мнемонической формулы, а повторное использование одной и той же формулы, должно рассматриваться как повторное использование пароля с точно такими же дополнительными рисками. Однако, при таком подходе теряется весь смысл использования мнемонических формул.

Таким образом, среди всех рассмотренных выше вариантов генерации надёжных паролей, наиболее предпочтительными во всех отношениях представляются мнемонические пароли, сгенерированные на основе несвязанного набора слов, или с помощью осмысленной парольной фразы. Очевидно, что в первом случае, формируется потенциально более стойкий пароль, хотя и обладающий гораздо худшими мнемоническими свойствами, так как беспорядочный набор слов в сравнении с осмысленной фразой запомнить гораздо сложнее, хотя и гораздо проще по сравнению с беспорядочным набором символов. В последнее время именно парольные фразы чаще всего рекомендуются использовать в целях повышения безопасности доступа к аккаунтам.

Недавно появились инициативы, подобные Fastwords, авторы которой уверяют, что благодаря T9, на сенсорной клавиатуре смартфона гораздо удобнее набрать несколько слов, чем пароль с буквами в разных регистрах, и уж тем более - цифрами и специальными знаками. Даже корпорация Google недавно опубликовала советы по составлению устойчивых к взлому паролей, в которых она рекомендует формировать пароли на основе осмысленных фраз для их простого запоминания. Так, например, пароль "2bon2btitq" получается из закодированной гамлетовской фразы "To be or not to be that is the question". По достаточно распространённому мнению, фраза всего из четырёх случайных слов, несёт в себе 44 бита энтропии, что гораздо больше, чем даже у очень сильного пароля, составленного из случайно выбранных символов.

Однако, специалисты исследовательской лаборатории компьютерной безопасности Light Blue Touchpaper в ходе многочисленных исследований пришли к выводу, что оценки энтропии парольных фраз, используемые в качестве показателя их устойчивости к взлому, в действительности обычно оказываются сильно оптимистически завышенными. Для проверки энтропии использовалась база фраз системы Amazon PayPhrase, которая в течение сравнительно непродолжительного времени работала для американских пользователей. Несмотря на сравнительно небольшой объём этой базы данных, полученные результаты всё же оказались весьма убедительными.

Для первого эксперимента исследователи составили словарь из 20000 фраз, используя названия спортивных команд, названия фильмов, список фраз из Википедии и список популярных фраз из словаря городского жаргона Urban Dictionary. Используя такой подход, им удалось взломать почти 8000 аккаунтов. Проведя соответствующие расчёты, авторы исследования получили эмпирические оценки энтропии идентификационных фраз на уровне 20 бит против хакера, который пытается скомпрометировать 1% существующих аккаунтов. Хотя этот показатель заметно лучше, чем у среднестатистического пароля, энтропия которого, в среднем, составляет не более 10 бит (при проведении оценки по такой же методике), но явно не удовлетворяет современным требованиям безопасности. Но главное, всё это очень плохо согласуется с традиционными теоретическими оценками для энтропии парольных фраз.

Наиболее убедительное объяснение полученных результатов заключается в пренебрежении пользователями правила использовать только случайные слова, являющегося в данной технике ключевым правилом. В дальнейшем исследователи провели лингвистический анализ слов в идентификационных фразах и выявили склонность пользователей к выбору парольных фраз вполне определённого типа, только с одним существительным, или только с одним глаголом. К тому же при составлении парольных фраз использовался очень ограниченный набор слов. Из-за этого реальная энтропия фразы оказывалась гораздо меньше своих ожидаемых оценок. Оказалось, что реальная энтропия парольных фраз, состоящих даже из 5 слов, обычно не превышала 30 бит, что в современных условиях уже не гарантирует надёжной защиты [8].

Очевидно, что результаты проведённых исследований можно в полной мере распространить и на мнемонические пароли, формируемые на основе парольных фраз и несвязанных между собой наборов слов. При этом дополнительно необходимо учитывать, что несмотря на избыточность языка, энтропия мнемонического пароля всё равно будет значительно меньше энтропии парольной фразы, по которой он сгенерирован. Эту особенность необходимо учитывать при определении минимальной длины парольной фразы.

2 Расчёт статистических характеристик паролей

Надёжность пароля является относительным понятием, так как зависит не только от сложности его случайного угадывания, но и от сложности верификации большого числа его предполагаемых вариантов. Во многих системах безопасности предусмотрена временная задержка, или полная блокировка доступа, если неверный пароль вводится несколько раз подряд. Это сильно затрудняет, или делает практически невозможным, взлом системы методом простого перебора, даже при использовании сравнительно простых паролей.

Однако, сама система авторизации вынуждена хранить информацию о пользовательских паролях для осуществления сверки с введёнными паролями. Чаще всего пользовательские пароли не хранятся в открытом виде, а шифруются с помощью специальных хэш-функций, непосредственное восстановление оригинального пароля по которым весьма затруднительно. Это делается из соображений безопасности, из-за невозможности гарантированного предотвращения доступа злоумышленников к этим данным. Более того, в ряде случаев, доступ к ним может быть получен сравнительно легко. Хотя наличие хэш-функции, само по себе, не открывает пароля, но позволяет верифицировать разные его варианты вне системы без каких-либо задержек и ограничений на число проверок. Именно в этой ситуации защищённость системы полностью определяется статистическими свойствами пароля [9].

Очевидно, что в рассматриваемом аспекте устойчивость пароля к взлому полностью определяется числом его возможных вариантов, и вероятностями выбора каждого из них, так как именно эти параметры определяют вероятность случайного угадывания пароля и среднюю длительность перебора наиболее перспективных вариантов при взломе. Однако, непосредственное использование этих параметров слишком неудобно, и на практике используется их обобщённая характеристика - информационная энтропия. Этот показатель получил очень широкое распространение в теории информации, изучающей фундаментальные свойства текстов, основы которой были заложены в середине XXв. американским инженером и математиком К. Шенноном.

Для оценивания количества информации, содержащейся в текстах, К. Шеннон предложил использовать меру, основанную на устранении исходной неопределённости, связанной с получением текста. При этом подходе понятия информация и неопределённость измерялись в одних и тех же единицах, и по сути отождествлялись. Примером количественной меры неопределённости случайного эксперимента дает теоретическая физика - такой мерой служит энтропия. Применительно к независимым испытаниям случайной величины ξ с распределением вероятностей $p_1 \dots p_n$ она определяется формулой

$$H(\xi) = - \sum_{i=1}^n p_i \log_x p_i \quad (1)$$

где x - основание логарифма, определяющее единицы измерения энтропии. При $x=2$ она измеряется в битах (наиболее распространённая на практике единица), при $x=3$ - в тритах, при $x=10$ - в хартли. При использовании в формуле (1) натурального логарифма ($x \approx 2.718$) энтропия измеряется в натах.

Распространённость измерения количества информации в Битах связана с так называемой теоремой кодирования [9], которая утверждает, что для кодирования любого, сколь угодно сложного, исхода достаточно всего двух символов, например 0 и 1. В этом случае длина полученного кодового слова будет сверху сколь угодно приближаться к значению $H(\xi)$. На основании этого минимальной единицей количества информации вполне естественно считать 1 бит. Из формулы (1) прямо следует, что для равновероятных событий, т.е. для $p_i = 1/n$ при любом $i = 1..n$, $H(\xi) = \log_2 n$. Кроме этого, для энтропии в общем случае имеет место следующее неравенство $H(\xi) \geq 0$, причём $H(\xi) = 0$ только в детерминированном случае, когда $p_i = 1$ для некоторого i , и $p_j = 0$ для всех $j \neq i$.

Мерой среднего количества информации, приходящейся на одну букву текста языка Λ (рассматриваемого как источник случайных текстов), служит величина H_Λ , называемая энтропией языка Λ . Для её оценивания Шеннон использовал последовательные приближения: $H_0, H_1, \dots, H_n$, где H_0 - предельная энтропия, определяемая размером алфавита, H_1 - энтропия позначной модели,

в которой p_i из (1) совпадает с вероятностью появления буквы S_i , в тексте.

Следующим, более точным приближением, является средняя энтропия вероятностного распределения биграмм, и т.д. В общем случае следует взять энтропию вероятностной схемы на r -граммах, деленную на r . Вычисления, проведенные для реальных англоязычных текстов дают следующие результаты $H_0 \approx 4,70$; $H_1 \approx 4,14$; $H_2/2 \approx 3,56$; $H_3/3 \approx 3,30$, и т. д. Исследования показывают, что с увеличением фрагмента r , отношение H_r/r стремится к некоторому пределу. Этот предел и принимается за определение энтропии H_Λ языка Λ :

$$H_\Lambda = \lim_{r \rightarrow \infty} \frac{H_r}{r} \quad (2)$$

При этом избыточность языка R_Λ определяет формула:

$$R_\Lambda = 1 - \frac{H_\Lambda}{\log_2 n} \quad (3)$$

Понятие "избыточность" было введено в связи с тем, что реальная информация, передаваемая символом текста, из-за наличия статистической взаимосвязи между символами, оказывается значительно меньше предельного значения, определяемого как $H_0 = \log_x n$ (где n - число букв в алфавите). Таким образом, величина $\log_x n - H_\Lambda$ характеризует неиспользованные возможности алфавита в передаче информации с помощью текста, а отношение

$$\frac{\log_a n - H_\Lambda}{\log_2 n} = 1 - \frac{H_\Lambda}{\log_2 n}$$

показывает, на какую часть букв можно было бы сократить естественный текст без потери содержащейся в нём полезной информации, за счёт применения оптимального кодирования. Из-за крайне ограниченных вычислительных возможностей, имевшихся в середине XXв., для оценивания отношения H_r/r в естественных текстов с позиции меры неопределенности опыта, Шеннон использовал методику, основанную на угадывании r -й буквы текста, при условии, что предшествующие его буквы известны [9]. Сложность угадывания r -й буквы оценивалась средним числом попыток, требующихся для нахождения верного ответа. Очевидно, что эти числа для разных значений r

определённым образом характеризуют статистическую структуру языка.

Шеннон произвел ряд подобных экспериментов, в которых r принимало значения 1, 15 и 100. При этом было обнаружено, что отгадывание сотой буквы по 99 предшествующим заметно более просто, чем угадывание 15-й буквы по 14 предыдущим. Опыты показали, что с ростом r отношение H_r/r убывает вплоть до 1.5 (при $r=30$), а дальнейший рост r уже не приводит к изменениям.

Пиотровским Б. Б. проводились аналогичные исследования для русского и французского языков [10]. Их результаты представлены в таблице 1.

Таблица 1 – Энтропия и избыточность языков

	Н _л , (бит на букву)		R _л , %	
	русский	французский	русский	французский
В целом	1,37	1,40	72,6	70,6
разговорный	1,40	1,50	72,0	68,4
литературный	1,19	1,38	76,2	71,0
деловой	0,83	1,22	83,4	74,4

Из приведенной таблицы видно, что естественные языки имеют весьма большую информационную избыточность. Это означает, что с помощью оптимального кодирования текста (например, при использовании кода Хаффмена и т.п.) его можно сжать до четверти длины без потери информации.

Следует также отметить, что существует и другой возможный подход к определению величины H_L для литературных текстов. Так, А. Н. Колмогоров, не согласившись с тем, что теоретико-информационные рассуждения игнорируют вопрос о смысловом содержании литературных текстов, предложил так называемый комбинаторный подход [9]. Согласно его представлениям, Шенноновскую энтропию H_L , приходящуюся на букву текста, можно определить тем условием, что для n -буквенного алфавита число текстов длины L , удовлетворяющих заданным статистическим ограничениям, равно (при достаточно больших L) не $n^L = 2^{L \log_2 n} = 2^{L H_0}$, как это было бы, если мы имели бы право брать любые наборы из L букв, а всего лишь $M(L) = 2^{L \cdot H_L}$.

По сути, это и есть асимптотика числа осмысленных текстов длины L для языка Λ . Исходя из этого, можно определить энтропию H_Λ формулой

$$H_\Lambda = \lim_{L \rightarrow \infty} \left(\frac{1}{L} \log_2 M(L) \right) \quad (4)$$

не зависящей от теоретико-вероятностных представлений. $M(L)$ в (4) можно оценивать подсчетом числа возможных продолжений литературного текста.

Пароли и парольные фразы являются достаточно короткими, поэтому средняя энтропия их символов значительно превышает среднюю энтропию языка H_Λ . В тоже время, их энтропия обычно значительно меньше H_0 и сравнивается с ней только при использовании случайного набора символов.

Конкретные значения энтропии символов пароля во многом зависят от способа его формирования. Если для этого используются слова естественного языка, то энтропия последующих символов всегда оказывается меньше чем у предыдущих. Увеличение количества информации при добавлении нового символа к набору определяется условной энтропией. Условная энтропия первого порядка определяет энтропию второго символа и вычисляется как

$$H_1(a | b) = - \sum_{i=1}^n p_i(a) \sum_{j=1}^m p_j(b | a_i) \log_x p_j(b | a_i), \quad (5)$$

где $p_i(a)$ - вероятности выбора вариантов первого символа a , $p_j(b|a_i)$ - вероятности выбора вариантов второго символа b при условии, что первым выбран a_i .

Аналогично вычисляется условная энтропия 2, 3 и n -го порядков. Однако, вычислительные формулы условной энтропии высших порядков весьма громоздки и неудобны в использовании. Вместо них может использоваться альтернативный способ вычисления условной энтропии, как разностей безусловных энтропий n грамм и $n-1$ грамм. Так, условная энтропия 1 порядка

$$H_1(a | b) = H_2(a, b) - H_1(a). \quad (6)$$

Для естественных русскоязычных текстов в литературе [10] приводятся следующие значения условных энтропий $H_0=5.0$; $H_1=4.36$; $H_2=3.52$; $H_3=3.0$.

Их можно использовать для ориентировочного оценивания энтропии коротких паролей, представленных русскими словами или словосочетаниями.

Сложность оценки реальной энтропии пароля связана с тем, что по нему самому невозможно точно определить ни мощность используемого при его составлении алфавита, ни вероятности выбора отдельных символов. Оптимистическая оценка энтропии, соответствующая предположению о равновероятных и независимых символах, имеет следующий вид

$$H_{\max} = L \cdot \log_2 N, \quad (\text{бит}), \quad (7)$$

пессимистическая оценка, исходящая из предположения, что пароль представлен фрагментом естественного текста, длиной более 30 знаков, равна

$$H_{\min} = 1.37 \cdot L, \quad (\text{бит}). \quad (8)$$

Реальная энтропия коротких паролей, представленных одним словом, определяется размером словаря и составляет от 10 до 20 бит в зависимости от распространённости используемого слова, и в первом приближении достаточно хорошо согласуется с пессимистической оценкой, определяемой по (8).

Усиление пароля за счёт манипуляций с регистром составляющих его символов позволяет увеличить энтропию не более, чем на L бит. В случае слабых паролей это даёт значительный эффект, но применительно к уже достаточно сильным паролям оказывается неэффективным. Аналогичный эффект даёт применение транслитерации, использование слов другого языка и т.п. По сути, все эти действия приводят к увеличению мощности используемого алфавита, что менее эффективно по сравнению с увеличением длины пароля. К тому же, такие усложнения сильно ухудшают запоминаемость пароля.

Таким образом, пароль, составленный на основе регулярных выражений естественного языка, при равной длине, имеет в 3-4 раза меньшую энтропию, по сравнению с паролем, составленным из случайно выбранных русскоязычных символов одного регистра. В таком пароле эффективно используется лишь четверть символов, а остальные символы являются балластом.

В свете вышеизложенного весьма привлекательной является идея ис-

пользования парольных фраз, составленных из специально подобранных слов, сочетающих в себе хорошую запоминаемость предложений естественного языка с высокой энтропией случайных наборов символов.

Произвольный выбор слов в парольной фразе позволяет обеспечить максимальную энтропию её первых символов. Однако энтропия последующих символов будет неизбежно меньше своего предельного значения, так как произвольный выбор нескольких символов, из-за объективно ограниченных размеров словаря, сильно сокращает варианты выбора остальных символов слова. Для вычисления условной энтропии отдельных слов русского языка использовался специальный словарь для модели `gensim ruscorpora-300`, которая может быть загружена в `python` из сети `internet` следующим образом

```
import gensim
import gensim.downloader as download_api
model = download_api.load('word2vec-ruscorpora-300')
```

Для определения объёма загруженного словаря, с дифференциацией по частям речи, может быть использован следующий программный код

```
n=[]; POS=['NOUN','VERB','ADJ','ADV']
for p in POS:
    q=0
    for k in range(len(model)):
        word=model.index_to_key[k]
        if word[-len(p):]==p: q+=1
    n.append(q)
```

Результаты выполнения этого кода показывают, что словарь содержит 128414 имён существительных, 2084 глаголов, 30221 прилагательных и 4842 наречия. Таким образом, загруженный словарь охватывает почти весь русский язык. Вычисления условных энтропий по этому словарю проводились с помощью функции `entropy()`, предназначенной для вычисления совместной энтропии N-грамм, вычисляющей двоичный логарифм числа уникальных N-грамм, для заданных частей речи POS, ограниченных максимальным числом символов `size_word`, в части словаря содержащей наиболее распространённые слова, размер которой задаётся параметром `part_dict`.

Кроме самой энтропии, функция *entropy()* вычисляет среднее число уникальных N-грамм, определяющее варианты выбора слов при конструировании предложения. Листинг этой функции представлен ниже

```
def entropy(N, POS, size_word, part_dict):
    # вычисление совместной энтропии набора N символов
    L=len(POS)
    s=set(); t=0
    for k in range(len(model)//part_dict):
        word=model.index_to_key[k]
        if word.find('.')==-1 and len(word)<=size_word+5 and word[-L:]==POS:
            if ord(word[0])>1071 and ord(word[0])<1107:
                s.add(word[:N])
                t+=1
    return np.log2(len(s)), t/len(s)
```

Сами вычисления выполнялись с помощью программного кода

```
H=[]; N=[]
for p in POS:
    h=[]; n=[]
    for k in range(1,7):
        e=entropy(k, p, 100, 1)
        h.append(round(e[0]-sum(h),2))
        n.append(round(e[1]))
    H.append(h)
    N.append(n)
```

Полученные результаты для разных частей речи, и разных, вплоть до 5-го, порядков условной энтропии, представлены в таблице 2.

Таблица 2 – Условные энтропии в отдельных словах

	H ₀	H ₁	H ₂	H ₃	H ₄	H ₅
Сущ.	5,00	4,66	3,29	1,80	0,96	0,46
Глаг.	4,81	3,75	2,00	1,44	0,82	0,41
Прил.	4,91	3,98	2,72	1,63	0,78	0,42
Нар.	4,81	3,53	2,12	0,99	0,48	0,1

Из представленной таблицы 2 видно, что безусловная энтропия первых символов слова близка к безусловной энтропии текста в целом. Условная энтропия первого порядка, определяющая количество информации во втором символе слова также близка, или даже превосходит H₁. Однако, с дальнейшим

увеличением порядка, в отличие от текста в целом, условная энтропия слова очень быстро уменьшается до нуля, так что в первом приближении, основная информация каждого слова заключена в первых трёх его символах.

Следует учитывать, что данные значения являются максимально возможными и соответствуют равновероятному выбору. В естественном языке вероятности появления разных букв неодинаковы, а между первыми символами соседних слов существует статистическая взаимосвязь. Поэтому в естественных русскоязычных текстах значения условных энтропий высших порядков оказываются даже меньше значений, приведённых в таблице 2.

В таблице 3 дополнительно представлены данные о среднем числе вариантов выбора слов, соответствующих произвольно заданной n -грамме.

Таблица 3 – Среднее число вариантов слов для заданной n -граммы

	n_1	n_2	n_3	n_4	n_5	n_6
Сущ.	3331	132	13	4	2	1
Глаг.	717	53	9	3	2	1
Прил.	986	62	9	3	2	1
Нар.	171	15	3	2	1	1

Из приведённых данных видно, что с увеличением числа произвольно заданных первых символов, число вариантов выбора слов резко сокращается, а составление из них осмысленного и хорошо запоминающегося предложения становится всё более сложным. Потому, при составлении искусственных предложений случайным образом целесообразно выбирать не более 2-3 первых символов в каждом слове. Так как, с одной стороны, это вполне достаточно для достижения близкой к максимально возможной энтропии слов, а с другой стороны - оставляет достаточно большой выбор слов, и обеспечивает возможность конструирования схожих с естественными, и хорошо запоминающихся предложений. Энтропия 4-го и последующих символов слова очень мала, и её увеличение путём случайного выбора символов уже не приводит к сколь-либо заметному приросту общей энтропии. К тому же, это приводит к получению во многом предопределённой фразы, в которой очень

сложно, или вовсе невозможно, установить дополнительные семантические связи, придающие мнемонические свойства естественного предложения.

Приведённые выше данные были получены без ограничений на максимальную длину слов и размеры словаря. Исследования влияния этих ограничений на условные энтропии показали, что ограничение максимального размера слов приводит лишь к незначительному снижению значений условных энтропий H_0 - H_2 , но резко ограничивает число возможных вариантов выбора слов. Причём, наиболее подверженными влиянию ограничений оказались глаголы и наречия, длину которых нецелесообразно ограничивать только до 8 символов. При построении предложения по случайным биграммам это обеспечивает выбор в среднем 15 разных слов. Для обеспечения такого же выбора для прилагательных, их длину можно ограничить до 7 символов, а в случае имён существительных - до 4 символов. Различное влияние ограничений на разные части речи очевидно связаны тем, что количества разных частей речи в словаре, равно как и в самом русском языке, очень сильно различаются.

В случае использования триграмм, чтобы обеспечить приемлемую для конструирования предложений гибкость, указанные выше границы максимальной длины слов должны быть дополнительно увеличены на 2-3 символа.

Из полученных результатов следует, что для получения максимальной энтропии в искусственном предложении предпочтительно использовать как можно больше существительных и как можно меньше глаголов и наречий.

Исследования влияния размеров словаря на условные энтропии проводились без дифференциации по частям речи. При этом размер словаря изменялся просто путём исключения части наиболее редко встречающихся в нём слов. Эти исследования показали, что сокращение словаря приводит к существенному уменьшению условных энтропий, при таком же существенном сокращении выбора вариантов слов. Удовлетворительные результаты удаётся получить только при использовании не менее 50% слов исходного словаря.

Ориентировочные расчёты показывают, что энтропия синтезированного таким образом предложения из 2 существительных, 2 прилагательных и гла-

гола длиной в 30 символов может достигать 60 бит, что в 1,5 раза больше энтропии среднестатистического русского литературного предложения.

3 Разработка схемы формирования парольной фразы

Из теории грамматики составляющих (phrase structure grammar) [11] следует, что всякая, сколь угодно сложная грамматическая единица состоит из более простых непересекающихся грамматических единиц. Согласно этому представлению, любое предложение естественного языка можно разделить на более простые компоненты, до отдельных слов. Причём, в любом языке существует лишь небольшое число устойчивых правил такого деления.

Синтаксический разбор преследует целью определение этих правил и отнесение разбираемого предложения к одной из разновидностей. Формирование искусственного предложения, в полной мере отвечающего всем этим правилам, представляет собой обратную задачу, являющуюся в целом более сложной, и требующей для решения введения целого ряда ограничений.

Самой крупной единицей, рассматриваемой в структурной лингвистике, является предложение (sentence), представляющее собой завершённое высказывание. Каждое предложение состоит из структурно обособленных клауз – синтаксических единиц, каждая из которых определяется одним сказуемым. Простое предложение состоит из единственной законченной клаузы (finite clause), а сложное – может состоять из нескольких, как финитных, так и нефинитных клауз, являющихся зависимыми частями (subordinate clause).

Отдельные клаузы сложного предложения объединяются с помощью синтаксических связей. В русском языке используется 3 их разновидности: подчинительные, сочинительные и бессоюзные [12]. Для образования подчинительной и сочинительной связей используются специальные подчинительные и сочинительные союзы, а также союзные слова. Иногда используются предложения сразу с несколькими, одинаковыми или разными, связями, в которых одновременно присутствует более 3-х составных частей.

Свойства завершенности и полноты предложение приобретает за счёт наличия семантической связи его отдельных составляющих. В отсутствии такой связи, даже синтаксически правильное предложение может оказаться бессмысленным, неестественным и, соответственно, трудно запоминаемым.

По мере усложнения искусственного предложения, вероятность случайного появления в нём семантических связей быстро уменьшается, а задача целенаправленного их формирования – очень сильно усложняется.

Пожалуй, самый простой путь усложнения предложения, позволяющий с высокой вероятностью получить осмысленное, и в то же время – хорошо запоминающееся, высказывание, заключается в использовании деепричастного оборота. Формально в русском языке, такие предложения рассматриваются как простые, осложнённые обособленным обстоятельством. Но по существу, они мало отличаются от сложноподчинённых предложений, а сам деепричастный оборот довольно часто рассматривается в роли инфинитивной зависимой клаузы. Положение деепричастного оборота, как и у обычного придаточного предложения, не фиксировано. Он может располагаться как в начале, так и в конце предложения, или между подлежащим и сказуемым.

Согласно правилам русского синтаксиса, деепричастный оборот является обстоятельством и обозначает добавочное действие. Поэтому для получения осмысленного высказывания в данном случае может быть достаточно наличия парной семантической связи между вершиной деепричастного оборота (деепричастием) и вершиной подлежащего (существительным), при условии правильного синтаксического согласования предложения. Заметного улучшения результата, при этом, можно добиться путём дополнительного попарного согласования вершин зависимых членов деепричастного оборота и дополнения основной клаузы (конечно, если эти члены присутствуют).

В случае же согласования обычных клауз сложного предложения установления попарных семантических связей между его отдельными словами, обычно недостаточно, и требуется установление семантической близости клауз в целом, что представляет собой гораздо более сложную задачу, к настоящему времени ещё не имеющую удовлетворительного решения [13]. К тому же, чрезмерное усложнение парольной фразы за счёт использования сложных предложений может быть нецелесообразным, так как даже несмотря на осмысленность, это приводит к ухудшению её мнемонических свойств.

Что касается отдельных клауз, то в их структуре выделяют 2 основные составляющие – именную и глагольную группы. В двухсоставных клаузах одновременно присутствуют обе эти составляющие, а в односоставных – только одна из них. В качестве парольной фразы предпочтительно использовать двухсоставные клаузы, так как в естественном языке они встречаются гораздо чаще односоставных, и поэтому проще запоминаются. К тому же они имеют устойчивую структуру (в повествовательном предложении сначала следует именная группа, а за ней – глагольная группа) и содержат в своём составе больше слов, что даёт возможность генерации более длинных паролей.

Именная группа состоит из вершины, в роли которой чаще всего выступает имя существительное в именительном падеже, и зависимых слов. В вершине именной группы может также находиться местоимение, однако использование местоимений в парольных фразах нежелательно, так как они обычно представлены короткими словами, и в русском языке их насчитывается всего около 50. Это ухудшает статистические характеристики пароля, так как ограничивает энтропию символов, сгенерированных на основе местоимений, и выбор остальных составляющих фразы при дальнейшем формировании пароля, в сравнении с почти неограниченными по размеру существительными, которых в русском языке [14] насчитывается более 100 тыс.

Именная группа может быть представлена только одним существительным, но для формирования как можно более длинного пароля её целесообразно дополнить зависимыми членами. Среди них в русском языке наиболее часто употребляются прилагательные в именительном падеже, согласованные с существительным по числу и роду. Прилагательные могут дополнительно модифицироваться наречием, что даёт возможность дополнительно увеличить размер высказывания. В современном русском языке насчитывается более 1000 наречий, которые могут начинаться почти на любую букву алфавита. Однако необходимость семантического согласования с прилагательным сильно ограничивает выбор, что ведёт либо к ухудшению статистических характеристик, либо - к неестественной плохо запоминающейся фразе.

В именной группе могут использоваться и другие зависимые члены, например относительные предложения, инфинитивные группы, предложные группы, именные адьюнкты, но их использование при формировании парольной фразы нецелесообразно по тем же причинам. В общем случае, любое дальнейшее усложнение именной группы приводит к резкому ухудшению её мнемонических свойств, что, по-видимому, связано со слишком редким использованием таких синтаксических конструкций в естественном языке.

Вершина одной именной группы, входящей в состав основной клаузы, обычно выступает в роли подлежащего предложения. Предшествующее ему прилагательное при этом выступает в роли определения. Остальные именные группы, которые также могут присутствовать в предложении, выступают совсем в других ролях. Так, например, именная группа в составе глагольной группы, выступает в роли дополнения. В этом случае, в отличие от подлежащего, находящееся в вершине существительное склоняется по падежам [12].

Глагольная группа, также как и именная, состоит из вершины и зависимых слов, но вершиной в данном случае является глагол. Разнообразие глагольных групп очень велико, так как один и тот же глагол, находящийся в вершине, может иметь разные наклонения, время, число, род и другие морфологические признаки. В синтаксически правильном предложении глагол иногда требует согласования с подлежащим по числу и роду. В аспекте задачи формирования парольной фразы предпочтительно использовать глаголы в настоящем времени. В этом случае их согласование по роду не требуется.

В русском языке присутствуют имена существительные, используемые только в единственном или только во множественном числе, что усложняет грамматическое согласование предложения. Самое простое и рациональное решение состоит в использовании глагола и существительного подлежащего только в единственном числе, с исключением из рассмотрения редко встречающихся слов, не имеющих формы единственного числа. Единственное число является нормальной формой, проще согласуется и запоминается. Это ограничение почти не влияет на статистические характеристики пароля.

В качестве зависимого слова глагольной группы, предшествующего глаголу, может использоваться наречие образа, меры или степени действия, носящее уточняющую функцию и выступающее в роли дополнения. Так как в русском языке насчитывается более 30 тыс. глаголов, и около 10 тыс. прилагательных, то использование наречий в данном случае может дать вполне приемлемый результат, как в отношении статистических характеристик пароля, так и в отношении мнемонических свойств парольной фразы.

Кроме этого, в распространённых предложениях, в состав глагольной группы входит зависимая именная группа, следующая сразу после глагола и выступающая в роли дополнения предложения. Сложность синтеза грамматически правильной глагольной группы заключается в том, что многие глаголы требуют вполне определённого падежа в зависимой именной группе, или даже использования предложной группы (состоящей из предлога и именной группы). Наличие предлогов в парольной фразе создаёт дополнительные затруднения, так как из-за слишком малого числа возможных вариантов, использование входящих в них символов в пароле ведёт к сильному ухудшению его статистических характеристик, а введение правила игнорирования предлогов вносит лишнюю путаницу и ухудшает мнемоничность пароля. К тому же, в доступных электронных словарях, в частности модуле `rumorphy2`, информация о релевантных падежах глаголов не содержится. Эта проблема решается дополнительной разметкой глаголов. Но это трудоёмкая задача. К тому же, предлог и именная группа требуют отдельного согласования.

Самое простое решение в данном случае заключается в использовании только переходных глаголов, требующих только винительного падежа без союзов и предлогов. Информация о переходности содержится в тегах `rumorphy2`, и её определение не требует дополнительной разметки словаря. В русском языке почти половина всех глаголов являются переходными, и введение данного ограничения может привести лишь к небольшому ухудшению характеристик пароля. С учётом существенного упрощения алгоритма генерации пароля, это решение может оказаться вполне приемлемым.

Структура деепричастного оборота во многом похожа на структуру глагольной группы. В роли зависимых членов в нём выступают предшествующее деепричастию наречие, и следующая после него именная группа, согласованная с деепричастием по падежу [12]. Здесь, также как и в глагольной группе, для упрощения алгоритма можно использовать только переходные деепричастия, требующие винительного падежа существительного без предлогов. Следует отметить, что по правилам русскоязычной лексики, в устной речи, использование деепричастных оборотов не рекомендуется, и они рассматриваются преимущественно как элементы печатных текстов. Это относится и к использованию сложных предложений. Однако использование длинного связанного предложения всё же оказывается предпочтительнее нескольких коротких предложений, как с точки зрения мнемоники, так и в плане сложности разработки алгоритма, предназначенного генерации фразы.

Обобщённая синтаксическая структура парольной фразы, разработанной с учётом рассмотренных выше особенностей, представлена на рисунке 1. С точки зрения русского синтаксиса это простое, полное, двухсоставное, распространённое, осложнённое обособленным обстоятельством предложение.

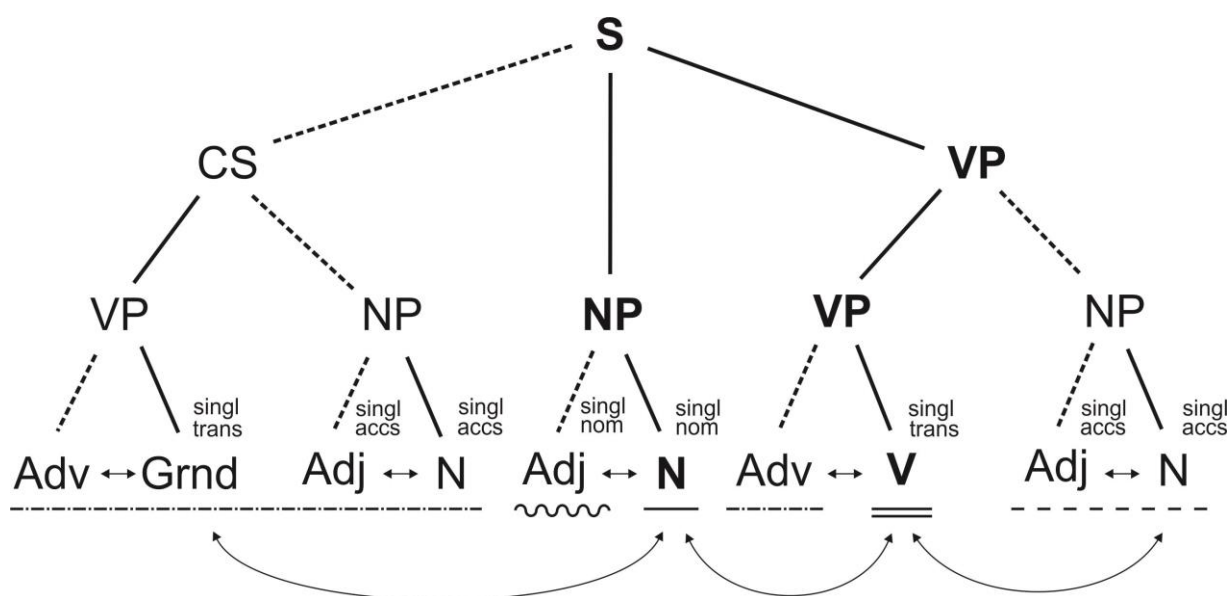


Рисунок 1 – Обобщенная синтаксическая структура парольной фразы
 NP – именная группа, VP – глагольная группа, CS – зависимая клауза, V – глагол,
 N – существительное, Grnd – деепричастие, Adv – наречие, Adj – прилагательное

На этой схеме сплошными линиями обозначены обязательные, а пунктирными линиями - опциональные структурные связи. Добавление или удаление опциональных связей позволяет изменять структуру предложения. При этом отсутствие опциональной связи означает отсутствие расположенного под ней структурного элемента и всех образующих составляющих. Таким образом, данная схема позволяет сформировать 84 типа предложений с разной синтаксической структурой. Самой длинное предложение, состоящее из 10 слов, получается при задействовании сразу всех опциональных связей. При полном отсутствии опциональных связей разработанная структура порождает самое короткое предложение, которое с точки зрения синтаксиса относящееся к категории двухсоставных нераспространённых предложений. Оно состоит всего из 2-х слов – существительного и глагола, выступающих в роли подлежащего и, соответственно, сказуемого (выделены жирным).

Благодаря возможности выбора структуры предложения разработанная схема обеспечивает высокую гибкость, и позволяет учесть индивидуальные требования и предпочтения конкретного пользователя. Это обеспечивает достижение оптимального соотношения надёжности пароля и мнемоничности соответствующей парольной фразы. Так как при отсутствии необходимости использования самой длинной фразы, целесообразно ограничиться выбором более короткой и лучше запоминаемой синтаксической структуры.

Для большей определённости на представленной выше структуре также обозначены некоторые морфологические признаки частей речи. В частности, все части речи используются только в единственном числе. Существительное подлежащего и прилагательное определения имеют именительный падеж, а существительные и прилагательные в обособленном обстоятельстве и дополнении – винительный падеж. При этом используются только переходные формы глагола и деепричастия, в единственном числе. Таким образом, при использовании данной схемы, для получения грамматически правильного предложения, требуется только согласование существительных и прилагательных по роду, на который никаких ограничений не накладывается.

Двухсторонними стрелками на рисунке 1 обозначены парные семантические связи, за счёт наличия которых формируемая парольная фраза приобретает свойства естественности и осмысленности. При условии синтаксически и морфологически правильной фразы, это позволяет достичь очень высоких мнемонических свойств, даже при довольно большой её длине.

Всего используется 8 семантических связей, три из которых связывают прилагательные с соответствующими существительными, а две – связывают глагол сказуемого с существительным подлежащего и существительным дополнения. Остальные связи установлены между подлежащим и деепричастием, деепричастием и соответствующим наречием, а также между глаголом сказуемого и вторым, характеризующим этот глагол, наречием.

Для определения семантического сходства может использоваться простой и достаточно эффективный подход на базе векторной модели слов. Согласно этому подходу, каждому слову приписывается определённый числовой вектор, а мера семантического сходства между словами вычисляется как расстояние между соответствующими им векторами. Традиционно, используется косинусное расстояние [14], характеризующее относительное сходство и позволяющее получить наиболее адекватные результаты сравнения слов.

По своей сути, этот подход основывается на использовании вероятностей совместной встречаемости пар слов в реальных текстах, а числовые векторы в компактном виде передают информацию об этих вероятностях. Непосредственное использование матрицы совместных вероятностей также возможно, но только для небольших словарей. В случае сотен тысяч слов её размеры оказываются настолько большими, что затрудняют обработку и хранение данных. Полнота передачи полезной информации векторами ограничена их размерностью, составляющей в современных моделях от 100 до 300.

Адекватность векторной модели во многом определяется содержанием массива реальных текстов, по которым она настроена. Некоторые массивы,

вместе с готовыми моделями, есть в открытом доступе. Для работы с векторными моделями доступны готовые библиотеки, например `gensim` для `python`.

С целью дальнейшего улучшения парольной фразы на неё могут быть наложены дополнительные ограничения. В первую очередь - на частоту встречаемости используемых слов. Так как предложения с редко встречающимися словами хуже запоминаются. Однако, чрезмерное сокращение словаря может привести к ухудшению статистических свойств пароля. Поэтому требуется определение оптимальных порогов отсечения для всех частей речи.

Вместе с этим могут быть установлены ограничения на максимальную длину используемых слов, предотвращающие получение слишком длинных предложений. Так как статистические свойства пароля определяются только первыми символами слов парольной фразы, а остальные символы носят второстепенную роль и увеличение их числа лишь ухудшает мнемонику фразы.

Очень большое значение имеет вопрос оптимального отбора символов парольной фразы для составления пароля, так как очевидно, что разные символы сформированной фразы с точки зрения информационной энтропии далеко неравноценны. Окончания слов слишком однообразны, и не могут иметь большую энтропию. Первым символом слова - напротив, с высокой вероятностью может быть почти любой символ алфавита (за исключением «ъ», «ь», «ы» и «й»). Это позволяет составить из них пароль с близкой к предельно достижимой для русского алфавита энтропией. Так как частота встречаемости символов в начале слова неодинакова, это в полной мере достигается путём предварительной стратификации вариантов выбираемой части речи по первому символу, с последующим случайным выбором одной из страт.

С помощью достаточно большого словаря аналогичным образом может быть составлен пароль из пар, или даже из троек, первых случайно выбранных первых символов слов, что позволяет удвоить, или утроить его максимальную длину. Хотя энтропия 2-го и 3-го символов оказывается на 10-20% и, соответственно, на 30-40% меньше, чем у первого, общая энтропия пароля, как и самой парольной фразы при этом существенно увеличиваются. Однако,

использование двоек, и тем более троек, случайных символов очень сильно усложняет семантическое согласование составляемой парольной фразы.

4 Разработка алгоритма формирования семантических паролей

Блок-схема алгоритма формирования семантического пароля представлена в приложении А. Работа алгоритма начинается с ввода размера базовых N-грамм (блок 2), используемого при последующем случайном отборе подходящих слов из общего словаря. Он может принимать значения 1, 2 или 3.

Далее (блоки 3-4), выполняется отбор уникальных N-грамм отдельно для существительных, прилагательных, переходных глаголов и наречий. При этом, отдельно для каждой части речи, накладываются дополнительные ограничения на максимальную длину слова (минимальная длина слова определяется выбранным размером N-граммы) и распространенность использования.

В модели *gensim* все слова упорядочены по распространённости в текстах, по которым она построена без учёта частеречной разметки. Поэтому для реализации дифференцированных ограничений, параллельно с отбором уникальных N-грамм подсчитывается количество слов с каждой N-граммой в словаре. Затем, в каждом из полученных наборов, удаляется столько последних N-грамм, чтобы обеспечить достижение заданного в % порога отсека.

Далее, для каждого из выделенных наборов вычисляется энтропия

$$H_k = \log_2 N_k, \quad (9)$$

где k - обозначает часть речи (существительное, переходный глагол, прилагательное, или наречие), N_k - размер набора N-грамм для заданной части речи.

Энтропия (9) достигается только при случайном выборе N-граммы из соответствующего набора. Поэтому, для отбора конкретных N-грамм для пароля используется генератор псевдослучайных числовых последовательностей (ГПСЧ), формирующий 10 случайных и независимых друг от друга чисел, каждое из которых находится в диапазоне, соответствующем размеру набора отбираемых N-грамм. На этом шаге, из отобранных наборов, сразу формируется пароль из $10N$ символов, и вычисляется его энтропия (блок 5) (10).

$$H = \sum_j^{10} \delta_j \log_2 N_{k(j)}, \quad (10)$$

где δ_j - наличие j -го слова в парольной фразе, $k(j)$ - соотв. ему часть речи.

Таким образом, на первом этапе данного алгоритма осуществляется генерация сильного пароля с гарантированно высокими статистическими характеристиками. Энтропия такого пароля отличается от предельно достижимой величины, определяемой согласно (7), лишь несовершенством используемого ГПСЧ, а также исходными и дополнительными ограничениями словаря, вводимыми с целью улучшения мнемоничности формируемых фраз.

На втором этапе алгоритма, по заданному (сгенерированному на первом этапе) паролю выполняется подбор наиболее релевантных и мнемоничных парольных фраз, а также оценивается общая энтропия парольной фразы. Для этого, по отобранным на первом этапе N-граммам, составляются наборы вариантов для каждого слова парольной фразы (блок 7), а также - наборы соответствующих этим вариантам числовых векторов. Очевидно, что с увеличением размера N-граммы, размеры этих наборов будут уменьшаться. Поэтому, как уже упоминалось, использование N-грамм более чем из 3-х символов нецелесообразно, так как в этом случае не будет, достаточной для составления естественного и легко запоминающегося предложения, свободы выбора.

Составление парольной фразы начинается с семантического согласования подлежащего, сказуемого и дополнения, в результате чего формируются простые предложения из 3-х слов. Так как в общем случае, в наборах отобранных слов, глаголов оказывается меньше чем имён существительных, то с целью увеличения разнообразия вариантов парольных фраз, для каждого глагола выбирается по несколько наиболее близких ему пар существительных. При этом, в качестве меры близости используется традиционная косинусная метрика, а в качестве критерия согласованности предложения используется максимальное значение среди 3-х парных расстояний. Таким образом удаётся выделить предложения наилучшим образом согласованные в целом.

Отбор наилучших вариантов предложений производится в тройном цикле (блоки 8-11), внутри которого для каждой тройки слов вычисляется критерий их согласованности, и в зависимости от его значения, тройки либо записываются в отсортированный набор, либо исключаются из рассмотрения.

Так как число возможных сочетаний троек слов может оказаться неприемлемо большим, то на размеры набора начальных предложений накладываются ограничения, таким образом, чтобы этот набор содержал только наилучшим образом согласованные начальные предложения. Полученный набор с высокой вероятностью будет содержать устойчивые словосочетания, часто встречающиеся в естественных текстах, использованных при построении векторной модели. Поэтому они будут отличаться хорошей мнемоничностью.

Главная проблема, возникающая при составлении осмысленных начальных предложений посредством векторной модели, заключается в необходимости дополнительного критерия оценки способности одного из пары выбранных существительных выступать в роли подлежащего, а другого - в роли дополнения. Мера близости, вычисляемая по векторной модели, не несёт никакой информации о порядке сравниваемых слов, а в случае модели, построенной по нормализованным текстам - ничего не говорит и об их падежах. Всё это может приводить к формированию абсурдных предложений, которые выражают невозможные для заданного сочетания субъекта и объекта действия.

Данная проблема может быть решена за счёт вычисления отдельных семантических векторов, одни из которых характеризуют имена существительные только в форме именительного падежа, когда они выступают в роли подлежащего, а остальные - в форме остальных падежей. С последующим использованием, одних векторов для вычисления меры сходства глагола и подлежащего, а других - для вычисления сходства глагола и дополнения. При этом, для пометки неизменённых существительных в нормализованных

текстах при построении семантической векторной модели `gensim` проще всего использовать верхний регистр первых символов в помечаемых словах.

Данный подход представляется весьма эффективным, однако его использование приводит практически к удвоению размеров исходного словаря, так как каждое существительное, на долю которых приходится основной объём, представляется дважды - как подлежащее и как дополнение. К тому же, готовые модели с такой специфической разметкой практически недоступны.

Альтернативный подход, возможно не столь эффективный, но не требующий построения специальных векторных моделей, основан на использовании возвратной формы глаголов. Очевидно, что объект воздействия обладает высокой степенью семантического сходства как с возвратной, так и невозвратной формами глагола. В то время как подлежащее предложения (субъект воздействия) близкое к невозвратной форме глагола, будет значительно дальше к его возвратной форме. В векторных моделях `gensim` возвратные глаголы, как правило, не нормализуются до невозвратных, и для них вычисляются отдельные семантические векторы. Это позволяет достаточно надёжно отличать подлежащее от дополнения даже с использованием существующих, широко доступных, векторных моделей.

Следует отметить, что библиотека `rumorphy2` не использует свойство возвратности глаголов и не позволяет непосредственно переводить их из одной формы в другую. Однако, в случае переходных глаголов, возвратная форма может быть получена простым добавлением к слову постфикса "ся". После чего, она может использоваться для вычисления косинусной метрики.

В разработанном алгоритме возвратные глаголы используются на этапе отбора семантически правильных предложений из первоначально выделенного набора. При этом, каждый глагол предложения переводится в возвратную форму, для которой вычисляется косинусное расстояние с существительным дополнения. Если вычисленное значение оказывается больше заданного порога, то предложение исключается из списка, так как имеющееся существительное, с точки зрения семантики, не может выступать в роли дополнения.

В результате такого отбора выделяется список предложений, существительные дополнений которых с высокой вероятностью могут выступать в роли объекта воздействия. При условии достаточной близости всех слов предложения, существительное подлежащего с высокой степенью уверенности может выступать в роли субъекта воздействия, а предложение в целом - является семантически согласованным. Поэтому расстояние между возвратным глаголом и существительным подлежащего не вычисляется.

На следующем этапе выполняется расширение начальных предложений списка путём подбора наиболее релевантных прилагательных для существительных подлежащего и дополнения и наиболее релевантного наречия для глагола (блоки 12-15). При этом для подбора используются соответствующие наборы частей речи, сформированные в начале алгоритма, на основе случайно выбранных N-грамм. В данном случае для получения хорошо согласованного предложения достаточно использовать соответствующие парные расстояния, так как семантическая связь добавляемых слов с остальными членами формируемого предложения мала и может не приниматься во внимание.

Дальнейшее расширение предложений списка выполняется за счёт добавления к ним деепричастных оборотов. Так как начальные предложения списка уже сформированы и семантически согласованы, это осуществляется путём последовательного добавления переходного деепричастия и зависимого существительного, с последующим расширением релевантными им наречием и прилагательным. В векторных моделях gensim деепричастия обычно нормализуются до глаголов, поэтому первоначально выполняется подбор наиболее релевантных глаголов, которые на заключительной стадии формирования предложения преобразуются в деепричастия. Здесь, как при формировании начальных предложений, возникает неопределённость объекта и субъекта добавочного действия, обозначаемого деепричастием. Поэтому для существительного подлежащего также вычисляются 2 косинусных расстояния, одно до исходного глагола, а другое до его возвратной формы.

В качестве наиболее релевантного выбирается наиболее близкий к подлежащему глагол, при условии, что дистанция между подлежащим и его возвратной формой имеет величину не меньше заданного порога. Конкретное значение оптимального порога подбирается экспериментальным путём таким образом, чтобы подлежащее предложения с высокой степенью надёжности выступало в роли субъекта добавочного действия, обозначаемого деепричастием. Таким путём, обычно, может быть достигнуто вполне удовлетворительное семантическое согласование расширенного предложения в целом.

В качестве зависимого существительного деепричастного оборота из соответствующего набора выбирается слово, имеющее минимальное расстояние как до исходного глагола, подлежащего переводу в деепричастие, так и до его возвратной формы. Благодаря этому гарантируется, что выбранное существительное может выступать в роли объекта добавочного действия.

Формирование предложений списка завершается добавлением наиболее релевантных деепричастию и зависимому существительному наречия и прилагательного. При этом, как и в основном предложении, семантическая связь добавляемых частей речи с остальными члена предложения практически отсутствует. В связи с чем, для их отбора вполне достаточно ограничиться использованием парных косинусных расстояний между соответствующими глаголами и наречиями, а также - существительными и прилагательными.

На данном этапе работы алгоритма формируется список синтаксически и семантически согласованных предложений. Однако все они остаются грамматически неверными, так как представлены нормальными формами частей речи, и для приобретения свойств естественного, хорошо запоминающихся предложений требуют дополнительного морфологического согласования.

Морфологическое согласование выполняется согласно обобщённой синтаксической структуре, представленной на рисунке 1, и заключается в установлении соответствующих ей падежей имён существительных, падежей и родов прилагательных, а также - в преобразовании глагола в деепричастие.

Все существительные дополнения и обособленного обстоятельства переводятся в винительный падеж. После этого определяется род каждого существительного, и все сопряженные с ними прилагательные переводятся в соответствующий род и падеж. Все соответствующие переходные глаголы переводятся в переходные деепричастия. После этого из слов полученного списка предложений удаляется частеречная разметка, и они выводятся на экран в удобном для восприятия виде. Так как все предложения списка соответствуют одному и тому же паролю, в качестве парольной фразы пользователь может выбрать любое из них, лучше всего запоминающееся конкретно для него.

Сгенерированная парольная фраза может использоваться не только для запоминания мнемонического пароля, представленного первыми буквами её слов, но и полностью. Это может быть целесообразным, например, при использовании неполной парольной фразы, полученной путём отбрасывания необязательных членов. В этом случае энтропия семантического пароля уменьшается и использование в качестве пароля сразу всей фразы позволяет частично или полностью восполнить это уменьшение. В этой связи значительный интерес представляет оценивание энтропии всей парольной фразы.

Известно множество приближенных формул, предназначенных для оценивания реальной энтропии парольных фраз [10], основывающихся на разных допущениях. В данном случае её разумно вычислять как сумму

$$H_{\Sigma} = H_p + H_r, \quad (11)$$

где H_p и H_r - энтропии семантического пароля и оставшихся символов фразы.

Основная сложность вычисления добавочного члена H_r связана с произвольным размером слов в парольной фразе. Самый простой способ ориентировочного оценивания H_r основывается на умножении количества символов парольной фразы, оставшихся после удаления из неё всех случайно выбранных N-грамм, на среднюю энтропию символа русскоязычных текстов.

Возможен и более точный способ, в котором проблема различия длин слов решается путём введения дополнительного символа "пробел", с помо-

щью которого длина слов парольной фразы могут быть выровнены. С учётом данных, представленных в таблице 2, в практических вычислениях достаточно использовать не более 5-6 оставшихся символов парольной фразы, так как оставшиеся символы, даже если они присутствуют, вносят лишь очень незначительный вклад в общую энтропию полученного предложения.

В отличие от энтропии короткого пароля, составленного из случайно отобранных N-грамм, вероятности появления символов в оставшейся части фразы могут существенно различаться, и в вычислениях следует использовать формулу (1). Ниже представлен листинг расширенной функции python предназначенной для вычисления энтропии, учитывающей эти особенности.

```
def entropy(N, POS, size_word, part_dict):
    # вычисление совместной энтропии набора N символов
    L=len(POS)
    s=set(); r={}; t=0
    for k in range(len(model)//part_dict):
        word=model.index_to_key[k]
        if word.find('.')==-1 and len(word)<=size_word+5 and word[-L:]==POS:
            if ord(word[0])>1071 and ord(word[0])<1107:
                s.add(word[:N])
                t+=1
                u=word[:-L]+" "
                u=u[:N]
                if u in r.keys():
                    r[u]+=1
                else:
                    r.update({u:1})
    p=list(r.values())
    hr=0
    for k in range(len(p)):
        hr+=np.log2(t/p[k])*p[k]/t
    return np.log2(len(s)), hr
```

Вычисления на словаре gisсoгpa-300, выполненные с помощью этой функции, показывают, что условные энтропии H_r , вычисленные с учётов вероятностей появления символов в естественных текстах, имеют значительно меньшие значения, чем приведённые в таблице 2 для специально отобранных N-грамм. Однако, с ростом порядка условной энтропии, убывают они гораздо медленнее. В пределе, при $N \rightarrow \infty$, можно заметить, что вычисляемая энтро-

пия слова стремиться к пределу, определяемому количеством слов в словаре.

Таким образом, общая энтропия парольной фразы определяется произведением количества входящих в неё слов и количества слов в словаре. Для фразы из 10 слов она может достигать 120 бит. Однако, это справедливо только для несвязанного набора слов. В случае осмысленных предложений между отдельными словами существует достаточно тесная взаимосвязь, и реальная энтропия таких фраз будет как минимум в 2 раза меньше, т.е. около 60 бит.

Из этого можно сделать вывод, что использование полных парольных фраз, сгенерированным по данному алгоритму нецелесообразно, так как практически вся их энтропии сосредоточена в первых символах слов.

5 Библиотеки python для работы с текстовыми данными

Библиотека морфологического анализа слов `pymorphy2` представляет собой, пожалуй, единственную доступную библиотеку подобного рода, пригодную для работы с русскоязычными (а также украинскими) текстами. Её установка осуществляется по сети `internet` с помощью следующей команды

```
pip3 install pymorphy2
```

установка оптимизированной версии (для этого может потребоваться настроенное окружение - компилятор C++ и т.п.) выполняется по команде

```
pip3 install pymorphy2[fast]
```

дополнительно, для работы может потребоваться обновление словаря

```
pip3 install -U pymorphy2-dicts-ru
```

Для выполнения морфологического анализа создаётся класс

```
import pymorphy2
```

```
morph = pymorphy2.MorphAnalyzer()
```

По умолчанию в работе `pymorphy2` используется русский словарь, однако, для исключения ошибок, словарь лучше указывать принудительно

```
morph = pymorphy2.MorphAnalyzer(lang='ru')
```

Каждый экземпляр созданного таким образом класса занимает около 15Мб памяти, поэтому не рекомендуется создавать большое их число.

Морфологический разбор слова осуществляется методом ***parse***

```
p=morph.parse('наполи')
```

В результате его выполнения возвращается многоэлементный объект разбора, каждый элемент которого содержит тег, нормальную форму разбираемого слова, оценку (`score`), а также тип используемого словаря, номер парадигмы и номер слова внутри неё. Несколько вариантов разбора получается потому, что одно и то же написание может соответствовать словам с разными смысловыми значениями, в том числе и разным частям речи. Оценка `score` определяет предполагаемую правильность разбора и вычисляется на основе вероятностей среднестатистического текста. В специализированных текстах значения `score` могут оказаться неадекватными, что следует учитывать. Элементы объекта разбора располагаются в порядке убывания `score`.

К каждому элементу объекта разбора можно обращаться по его индексу, например `r[0]`, или `r[2]`. Если разбираемое слово отсутствует в используемом словаре, то `rumorphy2` для его разбора задействует алгоритм автоматического предсказания. При этом, тип используемого словаря `methods_stack` будет иметь значение `<FakeDictionary>`, или что-то подобное. Контроль этого значения позволяет предотвратить формирование несуществующих слов.

Предсказатель основан на идее о том, что если русскоязычные слова оканчиваются одинаково, то с высокой вероятностью они имеют одинаковую форму. В `rumorphy2` реализованы 2 работающих совместно алгоритма предсказания. Первый основан на отсечении префиксов, а второй - на предсказании по хвосту. В первом случае, если 2 слова отличаются только тем, что к одному из них что-то приписано спереди, то, скорее всего, и склоняться они будут одинаково. Это особенно вероятно, если "X" - один из известных словообразовательных префиксов (например, "код" - "псевдокод"). В `rumorphy2` хранится небольшой список таких префиксов (например, "не", "анти", "псевдо", "супер", "дву" и т.д.), которые отсекаются, а остаток передается на дальнейший разбор. Если ни одного из таких префиксов не найдено, то анализатор отсекает от начала слова по одной букве, и проводить разбор полученного остатка, до получения правдоподобного результата.

Для того, чтобы предсказывать формы слов по окончаниям, при конвертации словарей строится DAWG со всеми возможными окончаниями слов (в последней реализации от однобуквенных до пятибуквенных) и каждому окончанию сопоставляется массив с возможными вариантами разбора слов с такими окончаниями. Схема хранения аналогична основному словарю, но вместо слов хранятся все их возможные окончания, а к номеру парадигмы и индексу формы в парадигме добавляется еще "продуктивность" правила - количество слов в словаре с данным окончанием, разбираемых данным образом. Для экономии места, все очень маловероятные правила отсекаются по специальным критериям. Сам разбор сводится к поиску наиболее длинной правой части разбираемого слова, которая есть в DAWG с окончаниями.

В русском языке продуктивными являются далеко не все части речи. В частности, нельзя приписать что-то к предлогу, с целью получения другого предлога. Все предлоги уникальны и занесены в словарь. По этой причине предсказывать незнакомые слова как предлоги будет некорректно. Такие, и подобные, некорректные варианты предсказания предсказатель игнорирует. Кроме этого, применяется большое количество других правил предсказания.

Разбор тегов в `rumorphy2` может выполняться по их атрибутам

`p[0].tag.POS`

определяет в роли какой части речи выступает слово в данном разборе. В случае отсутствия заданного атрибута, возвращается значение `None`. Кроме того, можно проверить, присутствует ли в теге набор заданных графем

`{'plur', 'past'} in p[0].tag`

`p[0].tag.POS=='plur'`

в результате выполнения такой команды возвращается `True` или `False`, а в случае ввода недопустимых графем возвращается ошибка.

По умолчанию графемы тегов в `rumorphy2` записываются в англоязычном варианте на латинице. Это не всегда удобно, поэтому присутствует дополнительная возможность вывода тегов в русскоязычном варианте

`p.tag.cyr_repr`

Англоязычные теги можно преобразовывать в русскоязычные

`morph.lat2cyr('NOUN,anim,masc plur,abl')`

и наоборот - из русскоязычных в англоязычные

`morph.cyr2lat('СУЩ,од,мр мн,тв')`

Для каждого элемента объекта разбора можно получить лексему слова

`lex=p[0].lexeme`

представляющую собой многоэлементный объект, состоящий из всех форм данного слова. Элементы лексемы имеют точно такую же структуру элементов, как и элементы объекта разбора `morph.parse`: тег, нормальная форма, `score`, `methods_stack`, парадигма, номер в парадигме. Склонение слова, то есть перевод его в другую форму лексемы, в библиотеке `rumorphy2` осуществляется с помощью метода `.inflect`, применяемого к объекту разбора.

Например, перевод слова в родительный падеж множественного числа

```
p[0].inflect({'plur', 'gent'})
```

Для перевода слова в нормальную форму можно использовать

```
p[0].normal_form
```

В этом случае возвращается только нормальная форма слова. Для получения всего элемента разбора нормальной формы следует использовать

```
p[0].normalized
```

Для согласования слова с числительными используется метод `.make_agree_with_number(N)`, где N числительное. Например

```
p[0].make_agree_with_number(5).word
```

Библиотека `rumorphy2` использует для работы специальные словари из открытого проекта `OpenCorpora` [1]. Эти словари специально оптимизированы с целью сокращения объёма хранимых данных и ускорения выборок из них отдельных слов. Все слова словаря объединены в лексемы – наборы всех форм одного слова. Первым в каждой лексеме следует лемма - нормальная форма слова, в частности, для существительного — это слово в именительном падеже и единственном числе. Каждое слово лексемы характеризуется тегом, с помощью которого и осуществляется морфологический разбор. Тег представляет собой набор характеристик слова, намываемых в терминологии `rumorphy` граммемами. Все возможные варианты грамем, для соответствующих атрибутов тега, в порядке их следования в теге, приведены ниже.

Таблица 4 – Нестандартные грамемы `rumorphy2`

граммема	значение
LATN	токен состоит из латинских букв ("foo-bar" или "Maßstab")
PNCT	пунктуация (например, ",", " или "!?" или "...")
NUMB	число (например, "204" или "3.14")
intg	целое число (например, "204")
real	вещественное число (например, "3.14")
ROMN	римское число (например, XI)

Таблица 5 – Часть речи (POS)

граммема	значение	пример
NOUN	существительное	пользователь
ADJF	полное прилагательное	плохой
ADJS	краткое прилагательное	плох
COMP	компаратив	хуже, похуже, ниже
VERB	личная форма глагола	захожу, заходит, заходил
INFN	инфинитив глагола	заходить, зайти
PRTF	полное причастие	прочитавший, прочитанная
PRTS	краткое причастие	прочитана
GRND	деепричастие	прочитав, рассказывая
NUMR	числительное	один, шесть, двенадцать
ADVB	наречие	хорошо, больно
NPRO	местоимение	он
PRED	предикатив	некогда
PREP	предлог	в
CONJ	союз	и
PRCL	частица	бы, же, лишь
INTJ	междометие	ой

Таблица 6 – Падеж (case)

граммема	значение	вопрос	пример
nomn	именительный	Кто? Что?	пользователь авторизуется
gent	родительный	Кого? Чего?	авторизация пользователя
datv	дательный	Кому? Чему?	сообщить пользователю
accs	винительный	Кого? Что?	пользователь вводит пароль
ablt	творительный	Кем? Чем?	защита паролем
loct	предложный	О ком? О чём?	сведения о пользователе
voc	звательный	при обращении	гражданин , проходите
gen2	второй родительный		банка сахару
acc2	второй винительный		записался в солдаты
loc2	второй предложный		лежит в шкафу

Таблица 7 – Вид (aspect)

граммема	значение	пример
perf	совершенный	передал
impf	несовершенный	передаёт

Таблица 8 – Одушевленность (animacy)

граммема	значение	пример
anim	одушевлённый	пользователь
inan	неодушевлённый	пароль

Таблица 9 – Род (gender)

граммема	значение	пример
masc	мужской	аудитор, проверил
femf	женский	система, обновила
neut	средний	озеро, затопило

Таблица 10 – Включенность говорящего в действие (involvement)

граммема	значение	пример
excl	не включён	пойдите
incl	включён	пойдёмте

Таблица 11 – Наклонение (mood)

граммема	значение	пример
indc	изъявительное	авторизуюсь
impr	повелительное	авторизуйтесь

Таблица 12 – Переходность (transitivity)

граммема	значение	пример
tran	переходный	передаёт сообщение
intr	непереходный	авторизуется

Таблица 13 – Залог (voice)

граммема	значение	пример
actv	<i>действительный</i>	авторизирующий
pssv	<i>страдательный</i>	авторизируемый

Таблица 14 – Число (number)

граммема	значение	пример
sing	единственное	пользователь
plur	множественное	пользователи
Sgtm	только единственное	
Pltm	только множественное	ножницы

Sgtm, Pltm не являются значениями p.tag.number

Таблица 15 – Лицо (person)

граммема	значение	пример
1per	1 лицо	авторизуюсь
2per	2 лицо	авторизуетесь
3per	3 лицо	авторизуется

Таблица 16 – время (tense)

граммема	значение	пример
pres	настоящее	передаю
past	прошрое	передал
futr	будущее	передам

Словарь OpenCorpora содержит всего около 5 млн. отдельных русских слов, объединённых приблизительно в 400 тыс. лексем. Из слов выделяются приставки (префиксы), основы (сте́мы) и окончания (хвосты). При таком подходе все лексемы могут сохраняться в виде однозначно определяющих их основ, так как все слова, входящие в лексему, имеют одну и ту же основу. Это позволяет существенно сократить объём памяти, занимаемой словарём, а также - упростить и ускорить доступ к его отдельным словам при работе в программе. Оставшаяся часть каждой лексемы словаря определяется правилом формирования всех входящих в неё слов, которое называемым парадигмой. По сути, парадигма представляет собой специальную таблицу префиксов, окончаний и тегов для всех слов лексемы. Парадигма позволяет про-склонять слово, т.е. получить по заданной основе все его возможные формы.

Для разных лексем, парадигмы очень часто повторяются, так что для 400 тыс. лексем словаря получается всего около 3 тыс. уникальных парадигм.

Для компактного хранения парадигм применяется числовое кодирование, в соответствии с которому префиксам, хвостам и тегам присваиваются уникальные номера, а сами они хранятся в отдельных списках. Это обеспечивает дополнительное сжатие данных. На программном уровне, парадигмы записываются в одномерные массивы, первыми в которых следуют номера хвостов, вторыми - номера тегов и последними — номера префиксов.

В исходном словаре OpenCorpora присутствует информация о связях между отдельными лексемами, в частности — между лексемами глагола и родственного ему существительного. Это позволяет осуществить склонение слов так же и по частям речи, т. е. приводить одни части речи к другим. Однако, в библиотеке `rumorphy2` такая детальная информация отсутствует, и все связанные лексемы просто объединены в одни большие лексемы.

Для хранения данных о словах в `rumorphy2` используется конечный автомат DAFSA и библиотека DAWG. При том, в библиотеке DAWG размещаются не сами слова, а строки формата «слово; разделитель; парадигма; форма слова в парадигме», т. е. слово, вместе с его допустимым морфологическим разбором. Конечный автомат кодирует эти строки с помощью графа, пример одного из фрагментов которого представлен на рисунке 2.

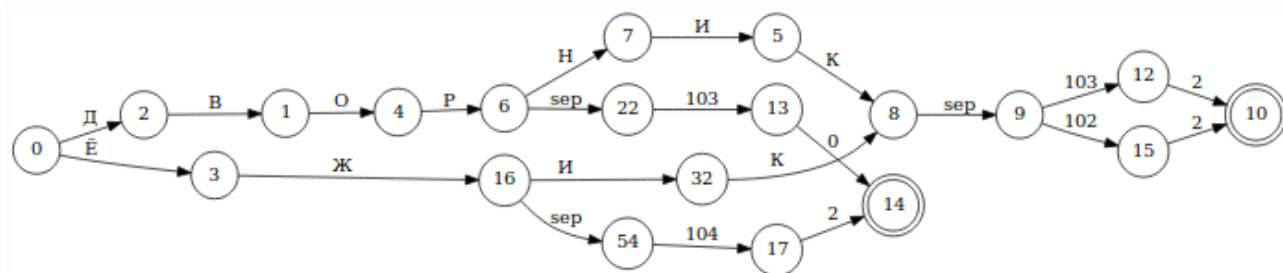


Рисунок 2 – Фрагмент графа DAFSA

Данный подход обеспечивает достаточно простую и эффективную работу со словарём, и позволяет существенно экономить память, так как общие части слов не дублируются при хранении, а DAWG хранит данные о словах и парадигмах в сжатом виде. При этом полностью хранятся все слова словаря, что полностью исключает возможность появления несуществующих слов.

При такой оптимальной организации, таблица с грамматической информацией (набор грамемм) занимает всего около 0,5 Мб машинной памяти. Парадигмы, и связанные с ними списки хвостов и префиксов — около 4 Мб, и сами слова в DAWG занимают около 7 Мб. Вместе со всеми сопутствующими данными `rumorphy2` занимает приблизительно 15 Мб памяти и обеспечивает скорость работы от 10 тыс. до 100 тыс. слов в секунду и даже более.

Библиотека `Gensim` предназначена для моделирования и обработки естественного языка (NLP), и в первую очередь - для анализа семантики текстовых документов. Она успешно применяется для классификации новостных сообщений, спама и другого недопустимого или нецензурного контента.

Установка библиотеки выполняется через `internet`, с помощью команды

```
pip3 install gensim
```

Данная библиотека всецело основывается на использовании методов машинного обучения, распознавания образов и нейронных сетей. Таким образом устанавливаются реальные семантические закономерности, присутствующие в текстах заданного тематического направления. Базовым элементом библиотеки `gensim` является семантическая модель, которая и устанавливает семантические связи между отдельными словами. `Gensim` использует разные модели представления текста, такие как `CountVectorizer`, `TfidfVectorizer`, `Word2vec`, `FastText` и `GloVe`. Первые 2 модели основаны только на частотах встречаемости слов и хорошо подходят для сравнения текстовых документов между собой. `CountVectorizer` по анализируемому документу строит словарь и вычисляет частоты встречаемости слов, в `TfidfVectorizer`, дополнительно, используется корректировка, снижающая важность часто встречающихся слов, что в целом, позволяет получить гораздо более адекватные результаты.

Однако, все вышеперечисленные языковые модели совершенно не подходят для анализа семантики отдельных текстов, особенно коротких, так как они никак не учитывают контексты слов, входящих в предложения.

Модель `Word2Vec` во много лишена этого недостатка. Она основана на

представлении слов анализируемого текста в виде N-числового вектора. Семантическая близость слов в этой модели определяется как расстояние между соответствующими им векторами в выбранной метрике. Сложность данного подхода заключается в том, что для адекватного представления взаимного соотношения слов в больших массивах осмысленных текстов требуются векторы очень большой размерности, причём, большинство элементов которых в любом слове оказываются нулевыми. Непосредственное хранение и обработка таких моделей требует неприемлемо больших вычислительных ресурсов. Поэтому исходные векторы этой модели кодируются по специальному алгоритму, что позволяет сократить их размерность до 100-300. Первую эффективную практическую реализацию концепции Word2Vec разработал Т. Миколов со своими сотрудниками в компании Google, реализовав процесс формирования векторов на основе обучения искусственной нейронной сети.

Модель FastText была разработана в компании Facebook и показала весьма высокие качество и скорость распознавания текстов. В отличие от Word2Vec, эта модель разбивает слова на несколько n-грамм, а векторное представление слова определяется суммой векторов для входящих в него n-грамм. Данная модель имеет преимущество в текстах с большим количеством орфографических ошибок и опечаток, так как способна всё равно правильно их классифицировать. Однако, это наиболее актуально только при мониторинге сообщений в социальных сетях и различных форумах internet.

Обе модели могут использовать 2 принципиально разных типа алгоритмов. В алгоритме Continuous Bag of Words получение числовых векторов основывается на предсказании слова по его контексту, а в алгоритме Skip-Gram, наоборот - на предсказании контекста по отдельным словам. Семантическая модель с алгоритмом Continuous Bag of Words в сравнении с Skip-Gram обеспечивает более высокую скорость обучения, но хорошо работает только для часто встречающихся слов, в то время как алгоритм Skip-Gram обеспечивает намного лучшее представление редко встречающихся слов.

Модель GloVe (global vectors for word representation) основана на алго-

ритме обучения искусственной нейронной сети без учителя, разработанном в стэнфордском университете. Она во многом похожа на модель Word2Vec, с тем отличием, что для извлечения семантических отношений между словами анализируемого текста используется матрица совместного использования, т. е. матрица совместных частот для разных пар слов. В реальных тестах эта семантическая модель оказывается самой эффективной, но вместе с этим, и самой ресурсоёмкой. Поэтому имеет ограниченное распространение.

Объект `wrd2vec` библиотеки `Gensym` создаётся следующим образом

```
model = gensim.models.Word2Vec()
```

все параметры этого объекта опциональные, т. е. если они не указаны в скобках, то используются нижеприведённые значения по умолчанию:

- `sentences=None` – набор предложений, может быть просто списком списков токенов, но для больших данных можно использовать объект, загружаемый с диска или по сети (например `BrownCorpus`, `Text8` или `LineSentence`);

- `corpus_file=None` – путь к файлу `corpus` в формате `LineSentence`. Этот аргумент используется вместо `sentences` (нельзя задавать оба аргумента одновременно, если ни один из них не задан, то созданная модель остается неинициализированной, и её нужно будет инициализировать в дальнейшем);

- `vector_size=100` – определяет размерность вычисляемых векторов,

- `window=5` – количество слов, рядом с анализируемым словом, используемых для вычисления числового вектора (размер контекстного окна);

- `min_count=5` – порог, слова с частотой ниже которого игнорируются;

- `max_vocab_size=None` – максимальное число слов в словаре, если число слов больше, то отбрасываются редко встречающиеся слова, используется при ограничениях оперативной памяти (на каждые 10 млн. слов требуется приблизительно 1 Гб памяти), значение `None` соответствует отсутствию ограничений;

- `sample=0.001` – порог, слова с частотой встречаемости выше которого будут отбрасываться с понижением частоты (выбирается в диапазоне от 0 до

1e-5);

- `seed=1` – начальное значение генератора случайных чисел.

- `workers=3` – количество ядер процессора, задействуемых для тренировки. Для воспроизводимости результатов нужно запускать модель только в одном потоке (`workers=1`), а в Python 3 ещё и использовать переменную окружения `PYTHONHASHSEED` для управления рандомизацией хэша;

- `min_alpha=0.0001` – определяет минимальную скорость обучения нейросети;

- `sg=0` – алгоритм обучения (Skip-Gram для `sg=1`, либо CBOW для `sg=0`);

- `hs=0` – если параметр равен 1, то для обучения модели будет использоваться иерархический softmax, если он равен 0, а выборка `negative` не нулевая, то при обучении будет использоваться выборка `negative`;

- `negative=5` – указывает сколько шумовых слов из выборки `negative` должно быть добавлено (обычно добавляется от 5 до 20 слов);

- `ns_exponent=0.75` – показатель, используемый при формировании `negative`;

- `cbow_mean=1` – при значении 1 используется средние векторы контекстных слов, при значении 0 — суммы векторов. Применяется только для CBOW;

- `hashfxn=<built-in function hash>` – хэш-функция, используемая для случайной инициализации весов с целью повышения воспроизводимости обучения;

- `epochs=5` – число эпох тренировки нейросети; `null_word=0`;

- `trim_rule=None` – правило обрезки словаря, используется для сокращения словарного запаса во время `build_vocab()` и не сохраняется как часть модели;

- `sorted_vocab=1` – если 1 то словарь сортируется по убыванию частоты слов;

- `batch_words=10000`, – целевой размер пакетов обучающих примеров (в словах), которые передаются отдельным рабочим потокам при обучении;
- `compute_loss=False` – если значение `True`, то вычисляется и сохраняется значение функции потерь, которое может быть получено с помощью специальной функции `get_latest_training_loss().maximum`;
- `callbacks=()` – последовательность обратных вызовов;
- `shrink_windows=True` – если значение `True`, эффективный размер окна равномерно выбирается из `[1, window]` для каждого целевого слова во время обучения, чтобы соответствовать приблизительному взвешиванию контекстных слов по расстоянию в исходном алгоритме `word2vec`.

При необходимости параметры модели можно проверить и изменить на другие значения следующим образом

```
print(model.workers)
model.workers=4
```

Для определения числа процессорных ядер нужно выполнить

```
import multiprocessing
n_cpu=multiprocessing.cpu_count()
```

Созданный объект в любое время может быть сохранён на диск

```
model.save("word2vec.model")
```

а потом - снова загружен в память для дальнейшего использования

```
model = gensim.models.Word2Vec.load("word2vec.model")
```

Для экономии памяти можно сохранять только словарь с векторами

```
model.wv.save("word2vec.wordvectors")
```

Для их загрузки используется следующая команда

```
wv = gensim.models.KeyedVectors.load("word2vec.wordvectors", mmap='r')
```

Их также можно загрузить из текстового файла

```
from gensim.test.utils import datapath
from gensim.models import KeyedVectors
wv=KeyedVectors.load_word2vec_format(datapath('file.txt'), binary=False)
```

а также из бинарного файла

```
wv= KeyedVectors.load_word2vec_format(datapath("file.bin"), binary=True)
```

Семантические векторы можно выделить из модели прямо в памяти

```
word_vectors = model.wv
```

а саму модель после этого удалить

```
del model
```

Для практического использования нужны только семантические векторы, которые занимают значительно меньше места, по сравнению с полной моделью. Однако, после их выделения, модель будет невозможно дообучить.

Основные функции. Получение вектора для слова:

```
vector = model.wv['наполь'])
```

получение списка N наиболее близких слов к слову:

```
sinonims = model.wv.most_similar('computer', topn=N)
```

получение векторов для набора индексов:

```
vector = model.wv[:10]
```

получение слов для набора индексов:

```
list(model.wv.index_to_key)[:10]
```

В сети internet есть примеры уже обученных моделей, информацию о которых можно получить с помощью следующей команды

```
print(list(gensim.downloader.info()['models'].keys()))
```

Эти модели могут быть загружены с помощью обычной команды

```
russian_model = gensim.downloader.load('word2vec-ruscorpora-300')
```

однако на самом деле, они содержат только семантические векторы и не могут дообучаться. Потому, обращение к функциям них выполняется напрямую, без использования свойства `wv`, присутствующего в полных моделях.

Если корпус обучающих данных был указан уже при создании объекта `Word2Vec`, то процесс обучения нейросети запускается автоматически, и в результате создаётся уже обученная модель. Если же он указан не был, то модель остаётся неинициализированной и необученной. В том случае нужно:

Создать словарь, для которого будут вычисляться векторы

```
model.build_vocab(sentences)
```

у этой функции один обязательный параметр `sentences` (обучающий корпус), или можно использовать `corpus_file`, остальные параметры опциональные:

– `update=False` – если значение `True`, то новые слова в предложениях

будут добавлены в исходный словарь существующей модели;

– `progress_per=10000` – количество обработанных слов перед каждым новым выводом хода выполнения операции;

– `keep_raw_vocab=False` – если значение равно `False`, необработанный словарь удаляется после завершения масштабирования, чтобы освободить память;

– `trim_rule=None` – усечение словаря (как и при создании модели).

Количество слов в модели можно узнать, выполнив

```
print(len(model.wv.vocab))
```

Для сброса весов нейросети перед обучением модели нужно выполнить

```
model.init_weights()
```

Запустить обучение нейросети

```
model.train(sentences, total_examples=model.corpus_count, epochs=model.epochs)
```

обязательными параметрами являются `sentences`, вместо которого можно использовать `corpus_file`, `total_examples` – полное число обучающих примеров, предложений, вместо которого можно использовать параметр `total_words = model.corpus_total_words` – полное число необработанных слов (иногда нужно указывать оба параметра), `epochs` — количество эпох обучения.

Все остальные параметры опциональные: `start_alpha` — начальная скорость обучения, `end_alpha` – конечная скорость обучения, `word_count` – число уже натренированных слов, произведение `workers*queue_factor` определяет размер очереди, `report_delay` — задержка сообщения о прогрессе (в секундах), `compute_loss` - Если `True`, то вычисляется и сохраняется значение функции потерь, которое может быть извлечено с помощью `get_latest_training_loss()`, `callbacks` - Последовательность обратных вызовов, которые должны выполняться на определенных этапах обучения.

Для нормализации векторов можно выполнить команду

```
model.init_sims(replace=True)
```

которая также освобождает память от исходных весов (при необходимости их сохранения нужно установить `replace=False`, только в этом случае оказывает-

ся возможным дообучить модель). В настоящее время эта команда считается устаревшей, и вместо неё рекомендуется использовать команду

```
gensim.models.keyedvectors.KeyedVectors.fill_norms()
```

Подготовка обучающего корпуса sentences играет очень важную роль в построении векторно-семантической модели, и во многом определяет качество получаемых результатов. Sentences представляет собой набор специально обработанных предложений исходного текста, который передаётся в функцию в виде вложенного списка слов формата utf-8. Каждый подсписок того вложенного списка содержит соответствующие слова предложения анализируемого текста. Самым первым этап подготовки исходного текста для обучения нейронной сети является его сегментация на отдельные предложения.

Известно, что русские предложения чаще всего заканчиваются точкой, восклицательным или вопросительным знаком. Такое простое деление среднестатистического текста по этим символам даёт около 7,6% ошибок. Ошибки 2 рода в данном случае возникают из-за того, что кроме знака конца предложения точка может использоваться в сокращениях, инициалах, списках. Ошибки первого рода могут возникать из-за того, что предложное вместо точки может завершиться многоточием, кавычкой в случае прямой речи, или каким либо знаком (особенно в сообщениях форумов и соцсетей).

Благодаря учёту этих особенностей, популярная библиотека NLTK позволяет уменьшить ошибки сегментации текста по предложениям до 5.7%. Для применения библиотеки, после её установки, загружаются разделители

```
import nltk  
nltk.download('punkt')
```

Загрузка текста осуществляется обычным образом

```
with open('file.txt') as f:  
s = f.read()
```

Затем выполняется сама процедура сегментации полученной строки

```
strings=nltk.sent_tokenize(s)
```

Известны и другие алгоритмы сегментации русского текста, так библиотека Razdel проекта Natasha допускает в среднем только 4,3% ошибок, а

библиотека Deerpavlov (достаточно ресурсоёмкая) – всего 4,1% ошибок.

Вторым обязательным этапом обработки текста является токенизация выделенных предложений, т. е. разделение их на отдельные слова. В токенизации предложений достаточно высокую эффективность показывает простое правило `[а-яё-]+|[0-9]+|[^а-яё0-9 ]`, дающее всего 1,9% ошибок. Ещё более эффективным является токенизатор `wordtokenizer` библиотеки NLTK, который можно применить к полученному массиву строк следующим образом

```
sentences=[]
for j in range(len(strings)):
    sentences.append(nltk.word_tokenize(strings[j]))
```

После токенизации, из полученных токенов как правило удаляются все символы, не являющиеся буквами или цифрами, а все буквы приводятся к нижнему регистру. То можно сделать с помощью следующего кода

```
for j in range(len(sentences)):
    for k in range(len(sentences[j])):
        s="".join(ch for ch in sentences[j][k] if ch.isalnum())
        sentences[j][k]=s.lower()
```

На следующем этапе обработки из всех токенов удаляются стоп-слова, представляющие собой различные предлоги, союзы, междометия, частицы и другие части речи, которые носят в основном служебную функцию и не несут никакой смысловой нагрузки, но в тексте встречаются очень часто.

Все стоп-слова как русского, так и других языков можно получить с помощью библиотеки `nltk`. После установки её нужно импортировать и загрузить стоп-слова для заданного языка из сети `internet`, следующим образом

```
nltk.download('stopwords')
words = nltk.corpus.stopwords.words("russian")
```

При необходимости, список стопслов можно расширять, например

```
words.extend([" "])
```

Само же удаление стопслов осуществляется следующим образом

```
for j in range(len(sentences)):
    s=[word for word in sentences[j] if word not in words]
    sentences[j]=s
```

После этого обучающий корпус текстов sentences уже вполне пригоден для обучения нейронной сети. Однако, существенного повышения качества обучения можно добиться за счёт дополнительного применения лемматизации и частеречной разметки, которая иногда также называется POS – разметкой. Лемматизация заключается в приведении всех слов обучающего корпуса к их нормальной форме, т. е. к единственному числу, именительному падежу и т. п. Это позволяет многократно сократить объём используемого словаря и существенно повысить прогностическую ценность обученной на нём модели за счёт удаления из обучающих данных мешающей информации. Частеречная разметка, помимо прочего, решает проблему одинакового написания разных частей речи. Кроме того, без частеречной разметки оказывается весьма затруднительно выполнить достаточно адекватную лемматизацию текста.

Для выполнения лемматизации и частеречной разметки может использоваться библиотека Rymorphy2. Соответствующий код имеет вид

```
for j in range(len(sentences)):
    for k in range(len(sentences[j])):
        p=morph.parse(sentences[j][k])
        sentences[j][k]=p[0].normal_form + '_' + str(p[0].tag.POS)
```

Преимущество такого способа заключается в том, что частеречная разметка в данном случае полностью совпадает с тегами Rymorphy2, что существенно упрощает дальнейшую обработку данных. Анализ применения данного программного кода к реальным текстам показывает, что для 12% слов результаты обработки оказываются неоднозначными. Причем, как в отношении частеречной разметки, так и в отношении лемматизации. Это связано с тем, что Rymorphy2 для большинства слов возвращает сразу несколько вариантов грамматического разбора, в которых они иногда могут выступать в роли разных частей речи. При случайном выборе из всех предоставленных вариантов, доля ошибочной разметки и лемматизации составит около 6%, но при учёте оценки правильности морфологического разбора score, предоставляемой Rymorphy2, следует ожидать снижения частоты ошибок до 2-3%. Учитывая, что точность разметки частей речи в современных алгоритмах со-

ставляет 96-98%, эти результаты являются вполне удовлетворительными.

Функция лемматизации заложена также и в библиотеке NLTK (для её использования нужно выполнить загрузку `nltk.download('wordnet')`), однако она не поддерживает русский язык и подвергаемый с помощью её обработки текст остаётся без изменений. Кроме этого, для лемматизации текстов часто применяется библиотека UDPipe 2.0. Данная библиотека основана на нейросетевых моделях, актуальная версия может быть загружена командой

```
md=wget.download('https://rusvectors.org/static/models/udpipe_syntagrus.model')
```

Сама библиотека устанавливается с помощью команды

```
pip3 install ufal.udpipe
```

Для её использования необходимо импортировать модули

```
from ufal.udpipe import Model, Pipeline
```

И выполнить следующие команды:

```
model = Model.load(md)
```

```
prline=Pipeline(model, 'tokenize', Pipeline.DEFAULT, Pipeline.DEFAULT, 'conllu')
```

```
res = prline.process(string)
```

Входной текст `string` подаётся в виде необработанной строки, которая может содержать несколько предложений. Результаты обработки так же возвращаются в формате строки, содержащей несколько подстрок, разделённых символом `'\\n'`. Подстроки `res` соответствуют токенам анализируемого текста и содержат токен в исходной форме, токен в нормальной форме, тег части речи и некоторую другую информацию. Извлечение необходимых данных из `res` в данном случае реализуется программно, функциями обработки строк.

Сравнение результатов лемматизации и частеречной разметки оказывается явно не в пользу UDPipe 2.0. Последняя даёт значительно больше ошибок, что сразу бросается с глаза. В частности, имена существительные часто ошибочно распознаются как глаголы и т. п. Это свидетельствует о недостаточной точности актуальной нейросетевой модели. Библиотека UDPipe 2.0. даёт возможность обучать свои модели, но это связано с рядом затруднений, главным из которых является низкое качество существующих обучающих выборок, в качестве которых используются размеченные вручную тексты.

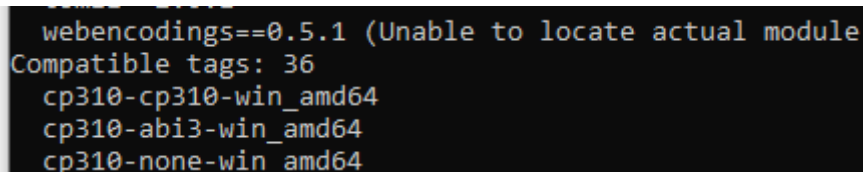
Следует отметить, что библиотека UDPipe 2.0 использовалась при подготовке обучающих корпусов проекта RusVectōrēs, в которых также наблюдаются вышеуказанные проблемы с POS-разметкой. Это создаёт серьёзные затруднения при работе с созданными в рамках этого проекта семантическими моделями. В составе самой библиотеки gensim для выполнения предварительной обработки текстов имеется функция `gensim.utils.simple_preprocess`, преимуществом которой является возможность чтения файлов по частям, что актуально при необходимости обработки очень больших массивов текста. Однако, эта функция реализует только токенизацию с очисткой исходного текста, и не позволяет выполнить ни частеречную разметку ни лемматизацию анализируемого текста. Поэтому её возможностей оказывается недостаточно.

6 Тестирование программы и разработка руководства пользователя

Программная реализация разработанного алгоритма генерации семантических паролей выполнена на кроссплатформенном языке программирования Python (версия интерпретатора 3.x). Листинг программы представлен в приложении Б. Основу данной программы составляет бесконечный цикл, в котором последовательно вызываются 3 функции и выводятся результаты работы программы. Функция `def gen_words()`, отбирает из модели `gensim` существительные, глаголы, прилагательные и наречия, начинающиеся на случайно сгенерированные наборы символов и соответствующие другим, приведённым в описании алгоритма, критериям отбора. Кроме этого, `def gen_words()` вычисляет энтропию для каждой части речи, для последующего подсчёта общей энтропии пароля. Функция `gen_sents()`, из сгенерированных наборов частей речи составляет синтаксически и семантически правильные предложения, а `morpho()` выполняет их морфологическое согласование.

Для использования программы в ОС Windows, требуется установка интерпретатора Python и библиотек `gensim`, `rumorphy2`, `numpy` и `random`. Интерпретатор для windows свободно распространяется в виде установочных файлов `.exe` или `.msi`. Его новые версии обычно не поддерживают старые ОС (для Windows7 без последних обновлений подходит интерпретатор не выше 7.7.6 версии). Важным нюансом, который необходимо учитывать при установке, является прописывание интерпретатора в переменной `PATH`. Если он в ней прописан, то скрипты `python` можно запускать из любого места системы. С одной стороны это удобно, но с другой - снижает защищённость системы перед вредоносным ПО. В новых установщиках имеется специальная опция добавления `python` в `PATH`. Старые версии интерпретатора, при необходимости, приходится прописывать в `PATH` непосредственно. Среди всех пакетов, наибольшие затруднения возникают при установке `gensim`. Сама её установка обычно проходит гладко, однако, попытки импорта в программе могут вызывать ошибки. Проблема заключается в совместимости `gensim`, `numpy` и `scipy`, не учитываемой при автоматической установке `gensim`.

Решение данной проблемы заключается в предварительной установке корректных версий numpy и scipy с последующей установкой gensim. В исследовании использовалась система Windows 10 Home 22H2 и интерпретатор python 3.10, после установки которого корректные версии библиотек определялись с помощью команды `pip debug --verbose`, результат вывода представлен на рисунке 3.



```
webencodings==0.5.1 (Unable to locate actual module  
Compatible tags: 36  
cp310-cp310-win_amd64  
cp310-abi3-win_amd64  
cp310-none-win_amd64
```

Рисунок 3 – Результат команды `pip debug --verbose`

Соответственно, корректные пакеты имеют следующие названия

`numpy-1.23.5+mkl-cp310-cp310-win_amd64.whl`

`SciPy-1.9.3-cp310-cp310-win_amd64.whl`

они доступны на сайте <https://github.com/cgohlke/numpy-mkl-wheels/releases>

Для ОС Windows 7 необходимый файл `numpy-1.21.0+mkl-cp37-cp37-win_amd64.whl` найти не удалось, а `gensim` запустить так и не получилось. Очевидно, что это относится и к более ранним версиям Windows. Однако, в перспективе можно полностью отказаться от использования `gensim`, сохраняя вместо этого заранее сформированный список слов `w`, и массив соответствующих им числовых векторов. Это сделает программу более универсальной, повысит её быстродействие и упростит подготовку интерпретатора.

Пакеты устанавливаются в командной строке из места их размещения

```
python -m pip install numpy-1.23.5+mkl-cp310-cp310-win_amd64.whl
```

```
python -m pip install SciPy-1.9.3-cp310-cp310-win_amd64.whl
```

после их установки, через internet, устанавливается библиотека `gensim`

```
python -m pip install gensim
```

Затем, устанавливаются остальные библиотеки (`random2` и `pymorphy2`)

```
python -m pip install random2
```

```
python -m pip install pymorphy2
```

После этого запускается интерактивный сеанс `python`

```
>python
```

и выполняется проверка корректности работы установленных библиотек

```
>import gensim, pymorphy2, numpy, random
```

отсутствие ошибок свидетельствует о корректности установки.

Затем, загрузчиком gensim загружается готовая модель ruskorpora-300, процесс загрузки показан на рисунке 4.

```
>>> import gensim.downloader as api  
>>> russian_model=api.load('word2vec-ruscorpora-300')  
[=====] 55.9% 111.0/198.8MB downloaded
```

Рисунок 4 – Процесс загрузки готовой модели ruskorpora-300

во избежание необходимости повторных загрузок, она сохраняется

```
russian_model.save('russian')
```

интерактивный режим интерпретатора python завершается командой exit(), после этого, интерпретатор полностью готов к работе с генератором паролей.

Возможно несколько вариантов запуска скрипта pythoh: в интерактивном режиме, в командной строке, или с помощью ярлыка. В первом варианте, сначала, запускается интерактивный режим

```
>python
```

в котором запускается сама программа генератора семантических паролей

```
>psswgen.py
```

во втором варианте, в командной строке сразу запускается команда

```
>python psswgen.py
```

в третьем варианте, программа запускается по щелчку на её файле.

Главное отличие этих вариантов заключается в том, что при запуске по ярлыку создаётся новое окно, в котором отображаются все результаты работы, и которое исчезает после её завершения. При запуске в командной строке результаты работы отображаются прямо в ней.

Для корректной работы файл программы должен находиться в одном каталоге с загруженными и сохранёнными моделями gensim. Для этого файла, как обычно, можно создать ярлык и разместить его на рабочем столе.

После запуска программы по ярлыку (это самый удобный способ), по-

является окно с предложением ввести размер N-граммы, который будет использоваться при формировании пароля, как это показано на рисунке 5.

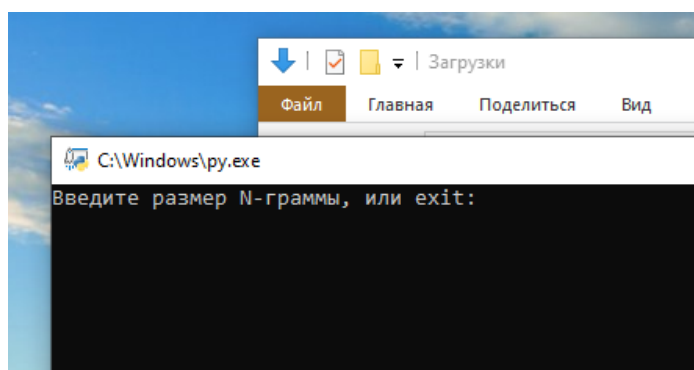


Рисунок 5 – Окно после запуска программы

После введения размера приблизительно через 30 секунд (время затрачивается на отбор подходящих слов из модели word2vec функцией `gen_word()`, в перспективе его можно существенно сократить) появляется предложение задать парольную фразу. Как это показано на рисунке 6.

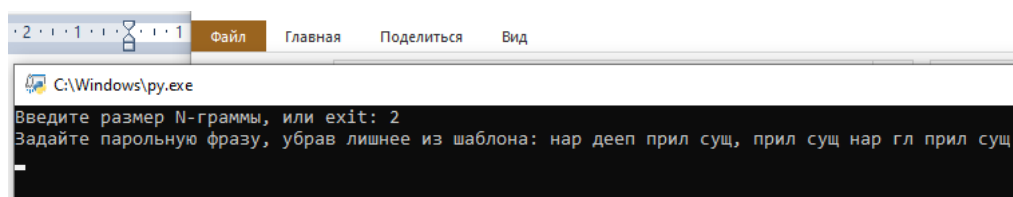


Рисунок 6 – Окно после ввода размера N-граммы

Структура фразы задаётся введением наименований входящих в неё частей речи (в том же порядке, как и в шаблоне) с отделением деепричастного оборота (при его наличии) запятой. Регистр ввода нижний. Могут вводиться как полные, так и сокращённые наименования, вплоть до первых букв. На рисунке 7 приведён пример полного ввода для предложения из 4 слов.

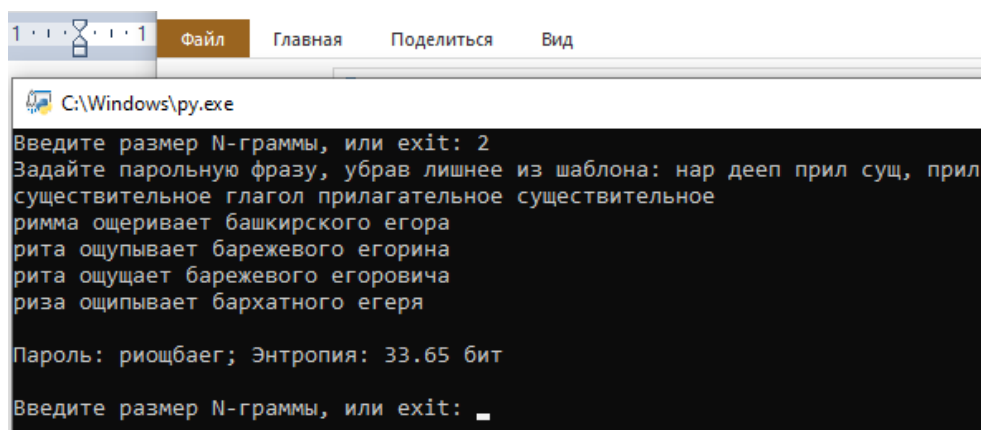
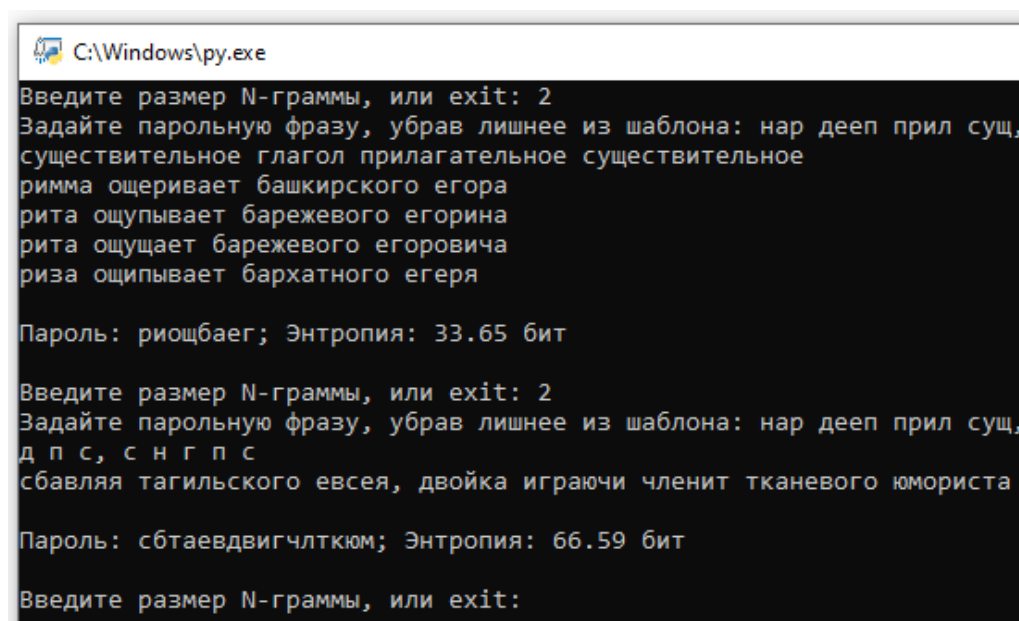


Рисунок 7 – Результаты работы программы

На этот раз результаты появляются почти без задержек и представляют собой 4 парольные фразы, пароль, составленный из первых двух символов каждого слова любой из них, и энтропия пароля. В данном случае получена энтропия около 34 бит, что уже намного лучше энтропии среднестатистического пароля. Данный пароль устойчив к взлому методом простого перебора.

В последней строке программа предлагает задать размер N-граммы для нового пароля. Это позволяет генерировать сразу несколько семантических паролей и соответствующих им парольных фраз, не выходя из сеанса. Следует отметить, что выбор пароля из нескольких сгенерированных вариантов в теории приводит к снижению его энтропии, но не более чем на $\log_2 N$ бит (где N - число сгенерированных вариантов). Это позволяет генерировать десятки вариантов различных паролей без существенного ущерба для энтропии.

На рисунке 8 представлены результаты для предложения из 8 слов.



```
C:\Windows\py.exe
Введите размер N-граммы, или exit: 2
Задайте парольную фразу, убрав лишнее из шаблона: нар дееп прил сущ,
существительное глагол прилагательное существительное
римма ощеривает башкирского егора
рита ощупывает бережевого егорина
рита ощущает бережевого егоровича
риза ощипывает бархатного егеря

Пароль: риощбаег; Энтропия: 33.65 бит

Введите размер N-граммы, или exit: 2
Задайте парольную фразу, убрав лишнее из шаблона: нар дееп прил сущ,
д п с, с н г п с
сбавляя тагильского евсея, двойка играючи членит тканевого юмориста

Пароль: сбтаевдвигчлткюм; Энтропия: 66.59 бит

Введите размер N-граммы, или exit:
```

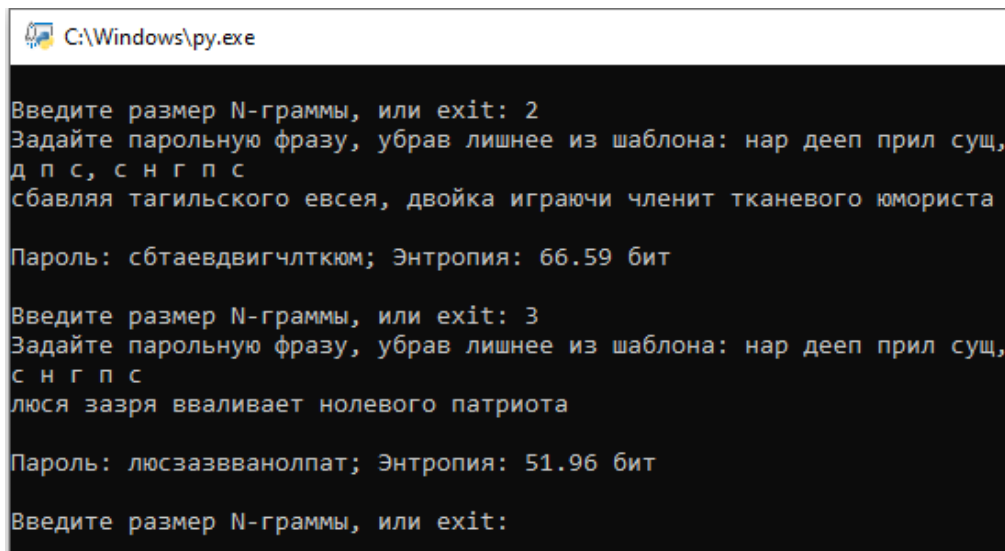
Рисунок 8 – Результаты генерации второго пароля

Видно, что энтропия пароля удвоилась, достигнув 66 бит. Пароль получился очень надёжным. Однако, для него программа смогла подобрать только 1 парольную фразу, которую, впрочем, довольно просто запомнить.

Количество подобранных программой парольных фраз от запуска к запуску может существенно различаться, так как одни наборы случайных чисел

позволяют подобрать много различных предложений, а другие - очень мало.

В целом, с увеличением размера N-граммы разнообразие парольных фраз уменьшается. На рисунке 9 подставлен результат генерации для N=3



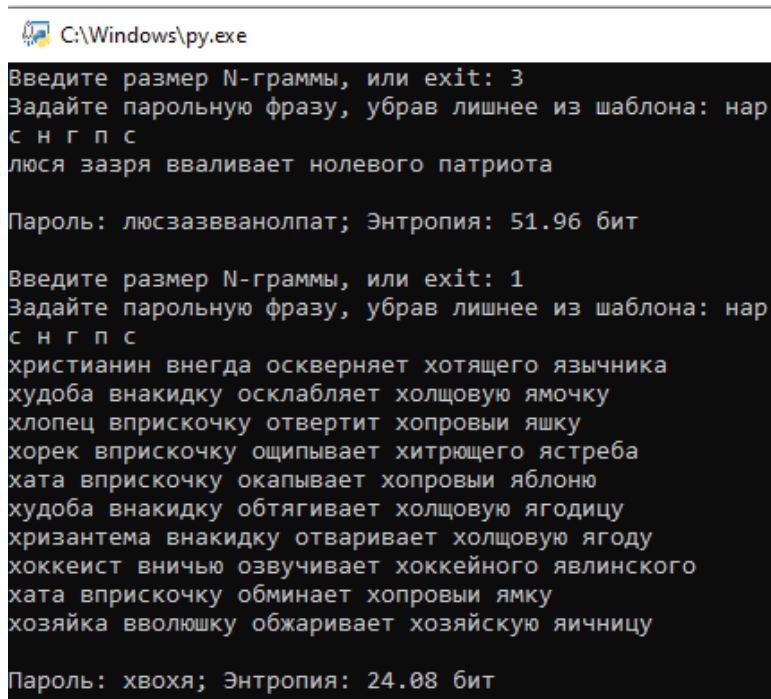
```
C:\Windows\py.exe
Введите размер N-граммы, или exit: 2
Задайте парольную фразу, убрав лишнее из шаблона: нар дееп прил сущ,
д п с, с н г п с
сбавляя тагильского евсея, двойка играючи членит тканевого юмориста
Пароль: сбтаевдвигчлткюм; Энтропия: 66.59 бит

Введите размер N-граммы, или exit: 3
Задайте парольную фразу, убрав лишнее из шаблона: нар дееп прил сущ,
с н г п с
люся зазря вваливает полевого патриота
Пароль: люсзазвванолпат; Энтропия: 51.96 бит

Введите размер N-граммы, или exit:
```

Рисунок 9 – Результаты генерации пароля по 3-граммам

Ниже, на рисунке 10, представлены результаты генерации для N=1



```
C:\Windows\py.exe
Введите размер N-граммы, или exit: 3
Задайте парольную фразу, убрав лишнее из шаблона: нар
с н г п с
люся зазря вваливает полевого патриота
Пароль: люсзазвванолпат; Энтропия: 51.96 бит

Введите размер N-граммы, или exit: 1
Задайте парольную фразу, убрав лишнее из шаблона: нар
с н г п с
христианин внегда оскверняет хотящего язычника
худоба внакидку ослабляет холщовую ямочку
хлопец вприскочку отвертит хопровый яшку
хорек вприскочку ошипывает хитрющего ястреба
хата вприскочку окапывает хопровый яблоню
худоба внакидку обтягивает холщовую ягодицу
хризантема внакидку отваривает холщовую ягоду
хоккеист вничью озвучивает хоккейного явлинского
хата вприскочку обминает хопровый ямку
хозяйка вволюшку обжаривает хозяйскую яичницу
Пароль: хвохя; Энтропия: 24.08 бит
```

Рисунок 10 – Результаты генерации по одиночным символам

Видно, что в последнем случае удаётся получить большое разнообразие парольных фраз, которые выглядят намного естественнее фраз, подбираемых при N=2, и тем более, фраз, подобранных при N=3. Однако, в этом случае достигается намного меньшая энтропия сгенерированного пароля.

В ОС Linux, работа с программой оказывается значительно проще. Интерпретатор python обычно уже входит в дистрибутив, а установка необходимых библиотек выполняется по internet терминальными командами

```
$pip3 install gensim
```

```
$pip3 install pymorphy2
```

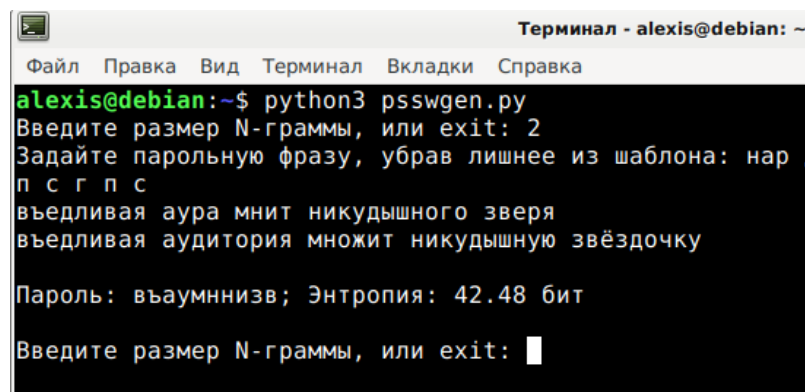
```
$pip3 install random2
```

Для исследования работы генератора семантических паролей использовался дистрибутив Linux Debian10 с окружением рабочего стола xfce. Скрипты python в нём запускаются либо из терминала, командой

```
$python psswgen.py
```

либо щелчком мыши на названии программы в файловом менеджере, или на ярлыке (который в xfce создаётся по аналогии с ОС Windows).

Пример запуска из терминала представлен на рисунке 11.

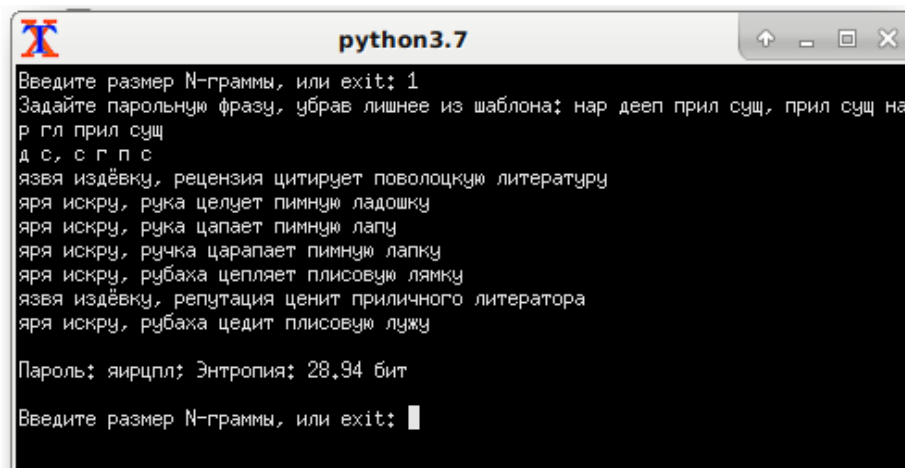


```
Терминал - alexis@debian: ~
Файл  Правка  Вид  Терминал  Вкладки  Справка
alexis@debian:~$ python3 psswgen.py
Введите размер N-граммы, или exit: 2
Задайте парольную фразу, убрав лишнее из шаблона: нар д
п с г п с
въедливая аура мнит никудышного зверя
въедливая аудитория множит никудышную звёздочку

Пароль: въаумннизв; Энтропия: 42.48 бит
Введите размер N-граммы, или exit: █
```

Рисунок 11 – Запуск программы в терминале Linux

Пример запуска с помощью ярлыка приведён на рисунке 12.



```
python3.7
Введите размер N-граммы, или exit: 1
Задайте парольную фразу, убрав лишнее из шаблона: нар дееп прил сущ, прил сущ на
р гл прил сущ
д с, с г п с
язвя издёвку, рецензия цитирует поволоцкую литературу
яря искру, рука целует пимную ладошку
яря искру, рука цапает пимную лапу
яря искру, ручка царапает пимную лапку
яря искру, рубаха цепляет плисовую лямку
язвя издёвку, репутация ценит приличного литератора
яря искру, рубаха цедит плисовую лужу

Пароль: яирцпл; Энтропия: 28.94 бит
Введите размер N-граммы, или exit: █
```

Рисунок 12 – Запуск программы в Linux по ярлыку

7 Вредные психофизиологические факторы, влияющие на психоэмоциональное состояние специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие

К вредным производственным факторам, в соответствии с действующим ГОСТ 12.0.003-2015 (система стандартов безопасности труда, опасные и вредные производственные факторы) относятся факторы, приводящие к заболеваниям, или к обострению уже имеющихся у человека заболеваний [15].

По характеру воздействия на здоровье работника все эти факторы делятся на 2 большие группы: факторы вызывающие хронические заболевания, а также влекущие обострение этих заболеваний, вследствие постоянного длительного низкоинтенсивного воздействия на организм, и факторы приводящие к острым заболеваниям, таким как отравления, травмы и другие поражение жизненно важных систем, не влекущие утраты трудоспособности, возникающие вследствие короткого, но - высокоинтенсивного воздействия.

По характеру своего происхождения все вредные и опасные (несущие риски стойкой утраты трудоспособности, инвалидизации, или смерти) подразделяются на 6 групп, среди которых, в отдельную группу выделены факторы, порождаемые психическими и психофизиологическими свойствами и индивидуальными особенностями работающего человека и его личности. Это субъективно плохое самочувствие, пребывание в состоянии сильного алкогольного опьянения, появление на работе под воздействием наркотических средств, состояния общей интоксикации, ярко выраженный абстинентный синдром, вынужденная потеря концентрации внимания работника и т.п. [16].

По своей природе, все вредные психофизиологические производственные факторы подразделяются на физические перегрузки, связанные с тяжестью выполнения работы и нервно-психические перегрузки, определяемые умственной напряженностью, возникающей при осуществлении трудовых обязанностей. В свою очередь, все физические перегрузки делятся на статические, связанные с необходимостью длительного принятия неудобной рабочей позы, динамические, связанные по большей части с весом груза, который

необходимо поднимать и перемещать в течение рабочего дня, и динамические нагрузки, связанные с повторением однообразных действий.

Для оценки условий труда и разработки комплекса мероприятий для минимизации негативного влияния физических перегрузок, их характеризуют с помощью ряда объективных показателей, таких как: величина динамической нагрузки, масса поднимаемого или перемещаемого груза, продолжительность однообразных движений, величина статической нагрузки, длительность нахождения в неудобной позе, частота вынужденных наклонов тела работника, и интенсивность вынужденных перемещений в пространстве.

Среди умственных перегрузок выделяют умственное напряжение, в том числе — связанное с чрезмерной информационной нагрузкой работающего, перенапряжение сенсорной системы анализаторов, в том числе — вызванное сильной информационной нагрузкой, монотонность и однообразие выполняемой работы, а также различные эмоциональные перегрузки [17].

С целью нормирования условий труда и организации мероприятий, снижающих негативное влияние вредных факторов нервно — психологической природы, также используются объективные показатели их выраженности. К ним относятся: продолжительность сосредоточенного наблюдения, длительность активного наблюдения за ходом протекания технологического процесса, количество объектов, за которыми работник вынужден вести одновременное наблюдение, интенсивность воспринимаемого информационного потока, в частности — количество звуковых и визуальных сообщений, принимаемых в единицу времени, интенсивность нагрузки на слуховой анализатор, интенсивность нагрузки на голосовой аппарат, а так же интенсивность нагрузки на зрительный анализатор при работе с различными оптическими приборами, буквенно-цифровыми индикаторами и мониторами [18].

Повседневная деятельность специалиста информационной безопасности, как правило, не связана с тяжелым физическим трудом, и практически все вредные факторы, действующие на него в течение рабочего дня связаны с

умственным перенапряжением, эмоциональными перегрузками, а также монотонным и однообразным характером выполняемой работы.

Показатель длительности сосредоточенного наблюдения принято оценивать в % от протяженности рабочей смены. По этому показателю выделяют 4 категории условий труда: до 25% - оптимальные условия труда (1 категория), 26-50% - предельно допустимые условия труда (2 категория), 51-75% - напряженный труд 1-й степени (3 категория) и более 75% - напряженный труд 2-й степени (4 категория). В зависимости от конкретной специализации и должности, условия труда специалиста по информационной безопасности по этому показателю могут существенно варьироваться, но обычно соответствуют 3 или 4 категории, т.е. напряженному труду 1-й или 2-й степени.

В силу своих профессиональных обязанностей, специалист по информационной безопасности значительную часть своего рабочего времени вынужден проводить за наблюдением объектов на весьма ограниченной площади монитора, с длительной фиксацией фокуса зрения. Без применения специальных мер предосторожности, это может приводить к сильному перенапряжению зрительного анализатора, а в долгосрочной перспективе — к развитию необратимых патологических процессов и расстройствам зрения [19].

Значительную часть времени пользователю приходится наблюдать за объектами на сравнительно небольшой площади экрана монитора. Это приводит к развитию сильного утомления в системе зрительного анализатора. Причиной этого являются фиксация близко расположенных, объединенных объектов, рассматривание мелких деталей, постоянный перевод взора с одного объекта на другой, частые резкие переходы от света к тени и обратно, пульсация освещенности и др. Все это через определенный период времени может привести к очень серьёзным патологическим изменениям органа зрения. Работа персонала за видеотерминалами требует большого нервно-психического напряжения, связанного с необходимостью длительного наблюдения концентрации памяти и внимания, решения сложных задач.

При работе в условиях зрительного напряжения происходят существенные изменения рефракции и аккомодационной способности глаз, ухудшается контрастная чувствительность, снижается устойчивость хроматического и ахроматического зрения, сокращаются границы поля зрения, снижаются острота зрения и скорость восприятия переработки информации. Весь этот комплекс изменений в сфере зрительного анализатора свидетельствует о развитии зрительного утомления. При чрезмерной продолжительности напряжения органа зрения утомление может накапливаться, приводить к развитию перенапряжения, возникновению серьёзных патологий зрительного аппарата.

Традиционно, для профилактики зрительных нарушений при длительной работе за монитором ПЭВМ рекомендуется введение обязательных регулярных перерывов для отдыха и приёма пищи (обеденный перерыв), перерывов для отдыха и личных потребностей, устанавливаемых с различной периодичностью, а также дополнительные короткие перерывы. По существующим нормам, общая продолжительность нахождения работника у крана ПЭВМ не должна превышать 6 часов за рабочую смену [20].

В последнее время, для операторов ПЭВМ разработана специальная система оптической коррекции, представляющая собой очки с набором светофильтров, обеспечивающие повышенный зрительный комфорта при наблюдении за цветными изображениями на расстоянии до 0,65 м.

Кроме этого, в профилактических целях, рекомендуется регулярный приём антиоксидантов, которые стимулирует восстановление зрительного пурпура, укрепляют соединительные ткани глазного дна и стимулируют синтез основного строительного материала сетчатки. Наиболее известным представителем таких соединений является бета-каротин, который может употребляться как в составе естественных продуктов, так и в виде фармацевтических препаратов, и различных пищевых биоактивных добавок. В последнее время, для тех же целей широкую популярность приобрели препараты на ос-

нове лютеин-зеаксантинового комплекса, обладающие выраженной антиоксидантной активностью и улучшающие метаболизм эпителия глаза.

Особенно большую роль в профилактике зрительного утомления играет предотвращение синдрома сухого глаза, связанного с пересыханием роговой оболочки вследствие недостаточно частого моргания. По имеющимся данным, тот синдром в той или иной степени присутствует более чем у 70% постоянных пользователей компьютеров. Для борьбы с этим нарушением рекомендуются специальные зрительные упражнения, а также приём химико-фармацевтических препаратов, предназначенных для устранения пересыхания оболочки глаза и гомогенизации структуры слёзной плёнки.

Помимо высоких зрительных нагрузок, работа специалиста по информационной безопасности связана с умственным перенапряжением, так как по роду деятельности он вынужден постоянно решать задачи различной сложности. Длительность умственного напряжения характеризуется долей рабочего времени, непосредственно связанного с умственным напряжением.

Условия труда по показателю умственного напряжения определяются по сложности решаемых задач в совокупности со скоростью их решения и длительностью умственного напряжения на протяжении всей рабочей смены. Интенсивность умственного напряжения обычно оценивается сложностью решаемых задач и временем, отводимым на их решение. Согласно руководству .2.2.2006-05 сложность умственного труда определяется такими факторами как: необходимость рассуждений, наличие ясно поставленной и отчётливо сформулированной цели, необходимость при решении задачи в построениях внутренних представлений о внешних событиях, наличие плана и инструкций по решению задачи, наличие иерархии подзадач, решение одних из которых основывается на решении предыдущих, известность последовательности действий и влияние её выбора на результат решения.

Как правило, по показателю сложность содержания работы, работа специалиста информационной безопасности относится к категории 3.2, так как осуществляется в условиях неполной или недостаточной информации, и

зачастую, для выявления новых угроз и разработки адекватных мер противодействия этим угрозам, требует применения эвристических приёмов.

Для профилактики умственного переутомления работающего персонала применяются самые разнообразные мероприятия. Известно, что самым эффективным средством для этого является сон, во время которого активизируются процессы регенерации нервных тканей и образования новых синаптических связей головного мозга. Однако, в силу ряда причин, использование сна в производственных условиях оказывается неприемлемым, и он как правило полностью происходит в свободное время человека, на которое работодатель повлиять никак не может, и этот вопрос находится исключительно в ведении самого работника. Это относится и к употреблению алкоголя, а также наркотических препаратов, оказывающих пагубное влияние на когнитивные способности персонала. Однако, работодатель не имеет в своём распоряжении сколь либо эффективных средств по противодействию этим занятиям.

Поэтому, для профилактики психосоматических нарушений, связанных с чрезмерными умственными нагрузками чаще всего, применяются разнообразные физкультурные методы, которые по своей эффективности находятся на втором месте после сна. В организации могут вводиться обязательные физкультурные пятиминутки, обязательное периодическое посещение тренажерных залов, организовываться разнообразные командные игры с выездом на природу, и т.п. Помимо этого, для предотвращения умственного переутомления показан приём различных ноотропных препаратов, активирующих метаболические процессы головного мозга, такие как пирацетам и т.п.

В целом, физическая культура рассматривается более предпочтительной по сравнению с приёмом фармацевтических препаратов, так как не связано с опасностью токсических осложнений. Умеренная физическая нагрузка показана практически всем категориям граждан независимо от пола и возраста. Она оказывает выраженное позитивное воздействие даже на лиц с хроническими заболеваниями и инвалидностью. Однако, важным условием полу-

чения положительного эффекта от физических упражнений является регулярность и достаточно большая продолжительность их выполнения. К тому же, большое значение имеет правильное дозирование нагрузок.

Заключение

В ходе выполнения настоящей дипломной работы были проанализированы актуальные требования, предъявляемые к паролям, а также основные подходы к формированию сильных и хорошо запоминаемых паролей. Проведён обзор методов вычисления информационной энтропии наборов символов различных паролей и коротких фрагментов текста естественных языков.

Разработана обобщённая синтаксическая структура парольной фразы, предназначенной для обеспечения хорошей запоминаемости пароля, состоящего из первых пар, троек, или отдельных символов образующих её слов. Данная структура позволяет автоматически формировать 84 типа синтаксически правильных русскоязычных предложений, состоящих от 2 до 10 слов.

Разработан алгоритм генерации семантических паролей, основанный на генерации случайных, независимых друг от друга символов, с последующим подбором для них грамматически правильной и семантически согласованной парольной фразы. Надёжность генерируемого пароля гарантируется случайной генерацией образующих его символов, а хорошая запоминаемость достигается за счёт получения хорошо согласованной, близкой к естественной, парольной фразы. Для семантического согласования предложений в работе использовалась векторная модель слов, показавшая хорошие результаты.

Разработанный алгоритм реализован в виде законченного консольного приложения на кроссплатформенном языке программирования python. Данное приложение было испытано в работе на операционных системах windows и linux, полностью подтвердив свою работоспособность. По результатам проведённых испытаний даны рекомендации по установке необходимых для работы программы библиотек, и по её практическому использованию.

В завершении работы проанализированы вредные психофизиологические факторы, влияющие на психоэмоциональное состояние специалиста по защите информации в процессе исполнения им служебных обязанностей, а так же определены физкультурно-оздоровительные мероприятия, направленные на предотвращение негативных последствий этих факторов.

Список использованных источников

1. Bugaj, Stephan Vladimir. More Secure Mnemonic-Passwords: User-Friendly Passwords for Real Humans” [Электронный ресурс] - URL: <http://www.cs.uno.edu/Resources/FAQ/faq4.html> (дата обращения - 5.04.2023)
2. Kotadia, Munir. Microsoft Security Guru: Jot Down Your Passwords [Электронный ресурс] - URL: <http://news.com.com/Microsoft+security+guru+Jot+down+your+passwords/2100-73553-5716590.html> (дата обращения - 12.04.2023)
3. Мнемонические формулы для запоминания паролей [Электронный ресурс] - URL: <https://www.securitylab.ru/analytics/354292.php> (дата обращения - 12.04.2023)
4. Придумать надежный пароль и запомнить его [Электронный ресурс] - URL: <https://habr.com/ru/post/528974/> (дата обращения - 15.04.2023)
5. Confidentialite et Transparence [Электронный ресурс] - URL: <http://www.fastword.me/> (дата обращения - 15.04.2023)
6. Stay safe online. Password [Электронный ресурс] - URL: <http://www.google.com/goodtoknow/online-safety/passwords/> (дата обращения - 15.04.2023)
7. АО «Лаборатория Касперского». Backdoor [Электронный ресурс] – URL: <https://encyclopedia.kaspersky.ru/knowledge/backdoor/>, (дата обращения - 15.04.2023);
8. АО «Лаборатория Касперского». Что такое Кейлоггер? [Электронный ресурс] – URL: <https://www.kaspersky.ru/blog/chto-takoe-keylogger/700/>, (дата обращения — 15.04.2023)
9. Яглом А. М., Яглом И. М. Вероятность и информация. - М.: Наука, 2020.
10. Алферов А. П., Зубов А. Ю., Кузьмин А. С., Черемушкин А. В. Основы криптографии Учеб. пособие, 2-е изд., испр. и доп. — М., 2020. — 480 с.

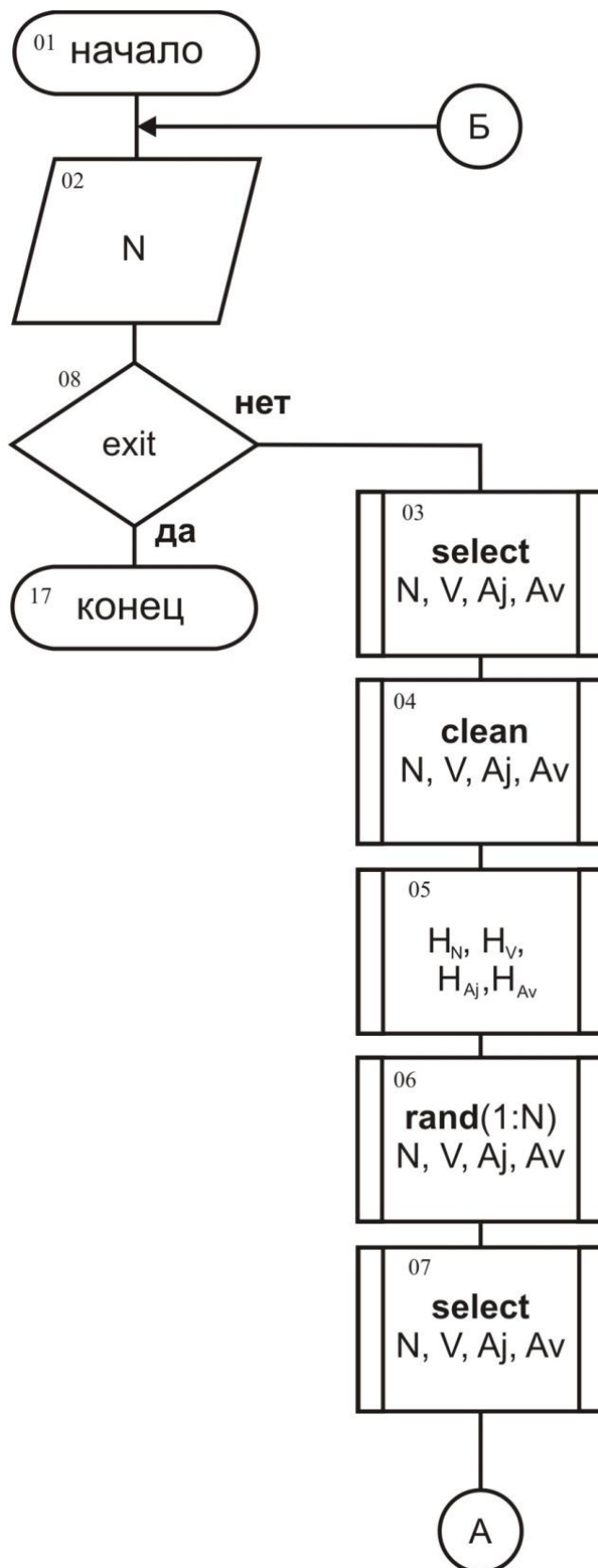
11. Шахматов, А. А. Синтаксис русского языка: монография / А. А. Шахматов. - 3-е изд., стер. - Москва: ФЛИНТА, 2020. - 714 с.
12. Лурия А. Р., Виноградова О. С. Объективное исследование динамики семантических систем. — «Семантическая структура слова». М., 2021.
13. Тестелец Я. Г. Глава II. Структура составляющих и фразовые категории // Введение в общий синтаксис. — РГГУ, 2021. — 800 с.
14. Апресян Ю. Д. Непосредственно составляющих метод // Лингвистический энциклопедический словарь / Гл. редактор В. Н. Ярцева. [Электронный ресурс] - URL: <http://www.tapemark.narod.ru/les/332a.html>. (дата обращения - 15.04.2023)
15. ГОСТ 12.0.003-2015 Система стандартов безопасности труда (ССБТ). Опасные и вредные производственные факторы. [Электронный ресурс] - URL: <https://marsbbz.ru> (дата обращения - 21.04.2023)
16. "Р 2.2.2006-05. 2.2. Гигиена труда. Руководство по гигиенической оценке факторов рабочей среды и трудового процесса. Критерии и классификация условий труда" (утв. Главным государственным санитарным врачом РФ 29.07.2005) [Электронный ресурс] - URL: <https://rusugleprof.ru/docs/OT42.pdf> (дата обращения - 21.04.2023)
17. Воробьева, М. А. Психология труда: учебное пособие для студентов и слушателей образовательных программ укрупнённой группы специальностей «Экономика и управление», «Менеджмент» и «Управление персоналом», а также для магистрантов / М. А. Воробьева; Урал.гос.пед.ун-т. — Екатеринбург, 2022. — 212 с.
18. Дружилов С.А. Психологические факторы здоровья человека и детерминанты его негативных психических состояний в труде// Международный журнал экспериментального образования. — 2021. — № 10-2. — С. 250-253;
19. Сухачев А.А. «Охрана труда в строительстве». Учебник для среднего профессионального образования. Издательство КноРус, 2011. Психофизиологические основы безопасности труда [Электронный ресурс] — URL: <http://www.kiout.ru/info/publish/15014>, (дата обращения — 17.05.23)

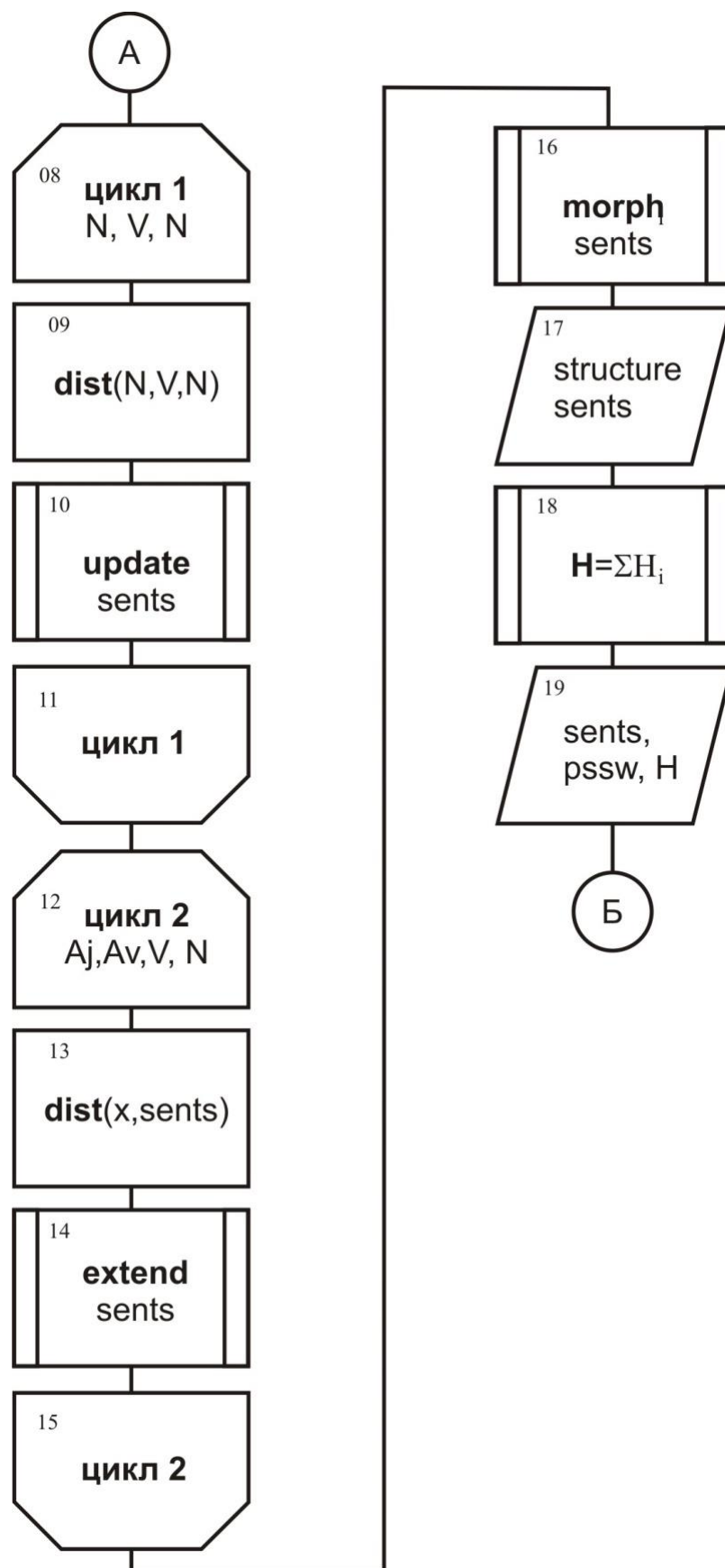
20. Девисилов В. Журнал «Охрана труда и техника безопасности на промышленных предприятиях». Психологические основы безопасности труда [Электронный ресурс] – URL: <http://coyt.org/index.php/news/233-psikhofiziologicheskie-osnovy-bezopasnosti-truda>, (дата обращения – 17.05.23)

Приложение А.

(обязательное)

Блок-схема алгоритма генератора семантических паролей





Приложение Б

(обязательное)

Листинг программы генератора семантических паролей

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Apr 26 12:51:06 2023
@author: martirosian
"""

import gensim, pymorphy2, random, numpy as np
morph = pymorphy2.MorphAnalyzer(lang='ru')
model = gensim.models.keyedvectors.KeyedVectors.load('russian')

def gen_words(N):
    # отбор частей речи из словаря
    noun=[]; verb=[]; adj=[]; adv=[]
    for k in range(len(model)):
        word=model.index_to_key[k]
        rus=True
        for j in word[:-5]:
            if ord(j)<1072 or ord(j)>1106: rus=False; break
        if not rus: continue # если не все символы русские и в нижнем регистре

        if word[-4:]=='NOUN' and len(word)<=10+5 and len(word)>3+5:
            p = morph.parse(word[:-5])
            if p[0].score>0.9: noun.append(word)

        if word[-4:]=='VERB' and len(word)<=10+5 and len(word)>3+5:
            p = morph.parse(word[:-5])
            if p[0].score>0.9 and p[0].tag.transitivity=='tran': verb.append(word)

        if word[-3:]=='ADJ' and len(word)<=10+4 and len(word)>3+4:
            p = morph.parse(word[:-4])
            if p[0].score>0.1: adj.append(word)

        if word[-3:]=='ADV' and len(word)<=10+4 and len(word)>3+4:
            p = morph.parse(word[:-4])
            if p[0].score>0.9: adv.append(word)

    # удаление реко встречающихся слов
    noun=noun[:int(0.2*len(noun))]
    verb=verb[:int(0.9*len(verb))]
    adj=adj[:int(0.9*len(adj))]
    adv=adv[:int(0.9*len(adv))]

    # отбор уникальных N-грамм
    nn=set()
    for j in noun:
        nn.add(j[:N])
```

```

h=list([np.log2(len(nn))]) # вычисление энтропии N-грамм существительных

vb=set()

for j in verb:
    vb.add(j[:N])
h.append(np.log2(len(vb))) # вычисление энтропии N-грамм глаголов

aj=set()
for j in adj:
    aj.add(j[:N])
h.append(np.log2(len(aj))) # вычисление энтропии N-грамм прилагательных

av=set()
for j in adv:
    av.add(j[:N])
h.append(np.log2(len(av))) # вычисление энтропии N-грамм наречий

# случайный отбор N-грамм и формирование списка слов
words=list([[],[],[],[],[],[],[],[],[],[]])
ngr=list(av)[random.randint(0,len(av)-1)]
for j in adv:
    if j[:N]==ngr: words[0].append(j)
ngr=list(vb)[random.randint(0,len(vb)-1)]
for j in verb:
    if j[:N]==ngr: words[1].append(j)
ngr=list(aj)[random.randint(0,len(aj)-1)]
for j in adj:
    if j[:N]==ngr: words[2].append(j)
ngr=list(nn)[random.randint(0,len(nn)-1)]
for j in noun:
    if j[:N]==ngr: words[3].append(j)
ngr=list(aj)[random.randint(0,len(aj)-1)]
for j in adj:
    if j[:N]==ngr: words[4].append(j)
ngr=list(nn)[random.randint(0,len(nn)-1)]
for j in noun:
    if j[:N]==ngr: words[5].append(j)
ngr=list(av)[random.randint(0,len(av)-1)]
for j in adv:
    if j[:N]==ngr: words[6].append(j)
ngr=list(vb)[random.randint(0,len(vb)-1)]
for j in verb:
    if j[:N]==ngr: words[7].append(j)
ngr=list(aj)[random.randint(0,len(aj)-1)]
for j in adj:
    if j[:N]==ngr: words[8].append(j)
ngr=list(nn)[random.randint(0,len(nn)-1)]
for j in noun:
    if j[:N]==ngr: words[9].append(j)

return words, h

```

```

def gen_sents(Ls, w):
# формирование Ls предложений из набора слов w
sents=[]
for i in range(Ls):
    q=0; s=[]
    for j in range(len(w[7])):
        word=w[7][j]
        reg=word[:-5]+"ся"+word[-5:]
        if reg in model:
            for k in range(len(w[9])):
                d=model.similarity(word, w[9][k])
                dr=model.similarity(reg, w[9][k])
                if d>q and dr>0.3*d:
                    q=d
                    s=[word, w[9][k]]
            if len(s)>0:
                sents.append(s)
                w[7].remove(s[0])
                w[9].remove(s[1])
            else:
                break
# добавить подлежащее
for i in range(len(sents)):
    q=0; s=[]
    for j in range(len(w[5])):
        word=w[5][j]
        d=model.similarity(word, sents[i][0])
        dr=model.similarity(word, sents[i][1])
        ds=model.similarity(sents[i][0], sents[i][1])
        if np.min([d,dr,ds])>q:
            q=np.min([d,dr,ds]); s=word
    sents[i].insert(0,s)
    sents[i].insert(0,q)
# добавить определение
for i in range(len(sents)):
    q=0; s=[]
    for j in range(len(w[4])):
        word=w[4][j]
        d=model.similarity(word, sents[i][1])
        if d>q:
            q=d; s=word
    sents[i].insert(1,s)
# добавить прилагательное в дополнение
for i in range(len(sents)):
    q=0; s=[]
    for j in range(len(w[8])):
        word=w[8][j]
        d=model.similarity(word, sents[i][1])
        if d>q:

```

```

        q=d; s=word
sents[i].insert(4,s)

# добавить обстоятельство
for i in range(len(sents)):
    q=0; s=[]
    for j in range(len(w[6])):
        word=w[6][j]
        d=model.similarity(word, sents[i][1])
        if d>q:
            q=d; s=word
    sents[i].insert(3,s)

# добавить деепричастие
for i in range(len(sents)):
    q=0; s=[]
    for j in range(len(w[1])):
        word=w[1][j]
        d=model.similarity(word, sents[i][2])
        if d>q:
            q=d; s=word
    sents[i].insert(1,s)

# добавить наречие
for i in range(len(sents)):
    q=0; s=[]
    for j in range(len(w[0])):
        word=w[0][j]
        d=model.similarity(word, sents[i][1])
        if d>q:
            q=d; s=word
    sents[i].insert(1,s)

# добавить существительное
for i in range(len(sents)):
    q=0; s=[]
    for j in range(len(w[3])):
        word=w[3][j]
        d=model.similarity(word, sents[i][2])
        if d>q:
            q=d; s=word
    sents[i].insert(3,s)

# добавить прилагательное
for i in range(len(sents)):
    q=0; s=[]
    for j in range(len(w[2])):
        word=w[2][j]
        d=model.similarity(word, sents[i][3])
        if d>q:
            q=d; s=word
    sents[i].insert(3,s)

```

```
return sents
```

```
def morpho(sents):
```

```
# морфологическое согласование предложения
```

```
for i in range(len(sents)):
```

```
    word=sents[i][0]; sents[i][0]=word[:-4]
```

```
    word=sents[i][6]; sents[i][6]=word[:-4]
```

```
    word=sents[i][1]
```

```
    p = morph.parse(word[:-5])
```

```
    sents[i][1]=p[0].inflect({'GRND'}).word
```

```
    word=sents[i][7]
```

```
    p = morph.parse(word[:-5])
```

```
    sents[i][7]=p[0].inflect({'3per'}).word
```

```
    word=sents[i][5]
```

```
    p = morph.parse(word[:-5]); g=p[0].tag.gender
```

```
    if g:
```

```
        word=sents[i][4]
```

```
        p = morph.parse(word[:-4]); u=p[0].inflect({g})
```

```
        if u: sents[i][4]=u.word
```

```
    word=sents[i][5]; sents[i][5]=word[:-5]
```

```
    word=sents[i][3]
```

```
    p = morph.parse(word[:-5]); g=p[0].tag.gender
```

```
    sents[i][3]=p[0].inflect({'accs'}).word
```

```
    if g:
```

```
        word=sents[i][2]
```

```
        p = morph.parse(word[:-4]); u=p[0].inflect({g,'accs'})
```

```
        if u: sents[i][2]=u.word
```

```
    word=sents[i][9]
```

```
    p = morph.parse(word[:-5]); g=p[0].tag.gender
```

```
    sents[i][9]=p[0].inflect({'accs'}).word
```

```
    if g:
```

```
        word=sents[i][8]
```

```
        p = morph.parse(word[:-4]); u=p[0].inflect({g,'accs'})
```

```
        if u: sents[i][8]=u.word
```

```
return sents
```

```
while True:
```

```
# бесконечный цикл
```

```
st=input("Введите размер N-граммы, или exit: ")
```

```
if st=="exit":
```

```
    break
```

```
else:
```

```
    N=int(st)
```

```
    w, h = gen_words(N)
```

```
    sents = gen_sents(10, w)
```

```
    sents.sort(reverse=True)
```

```
    for i in range(len(sents)): sents[i].pop(0)
```

```
    sents=morpho(sents)
```

```

s='нар дееп прил сущ, прил сущ нар гл прил сущ'
st=input("Задайте парольную фразу, убрав лишнее из шаблона: '+s+'\n')+ ' '
# отбор первых символов
j=st.find(",")
if j!=-1: st=st[:j].upper()+(st[j+1:])
i=1; j=st.find(" ")
while j!=-1: st=st[0:i]+st[j+1:]; j=st.find(" "); i+=1
# расшифровка введенной структуры
nums=[]
dc={'H':0, 'Д':1, 'П':2, 'С':3, 'п':4, 'с':5, 'н':6, 'г':7}
for i in st:
    if i in dc:
        nums.append(dc[i])
    else:
        continue
if 7 in nums:
    if nums[nums.index(7)+1]==4 and nums[nums.index(7)+2]==5:
        nums[nums.index(7)+1]=8; nums[nums.index(7)+2]=9
    elif nums[nums.index(7)+1]==5:
        nums[nums.index(7)+1]=9
    else:
        continue
else:
    continue
if len(nums)>10: continue
mask=10*[False]
mask[nums[0]]=True
for i in range(1,len(nums)):
    if nums[i]>nums[i-1]:
        mask[nums[i]]=True
    else:
        mask=10*[False]
        break
if not True in mask: continue

# вывод результатов на экран
nh=[3, 1, 2, 0, 2, 0, 3, 1, 2, 0]
for k in sents:
    if mask[3]:
        k[3]+=", "
    elif mask[1]:
        k[1]+=", "
psw=""; hp=0
for i in range(10):
    if mask[i]:
        psw+=sents[0][i][:N]; hp+=h[nh[i]]
for k in sents:
    s=""
    for i in range(10):
        if mask[i]:
            s+=k[i]+" "

```

```
print(s)

print("\nПароль: "+psw+"; Энтропия: "+"{:.2f}".format(hp)+" бит\n" )
```