ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования

Национальный исследовательский университет «Высшая школа экономики»

Факультет гуманитарных наук
Образовательная программа
«Фундаментальная и компьютерная лингвистика»

Феоктистова Эмма Александровна

ОТ НЕЙРОСЕТЕВОГО АНАЛИЗА К СЛОВАРНОМУ: МЕТОДЫ РАЗРАБОТКИ МОРФОЛОГИЧЕСКИХ АНАЛИЗАТОРОВ НА ОСНОВЕ ДАННЫХ, РАЗМЕЧЕННЫХ НЕЙРОСЕТЬЮ

Выпускная квалификационная работа студента 4 курса бакалавриата группы БКЛ191

Академический руководитель образовательной программы
канд. филологических наук, доц.
Ю.А. Ландер

_____

« » _____ 2023 г.

Научный руководитель

Профессор
О. Н. Ляшевская

Москва 2023

ОГЛАВЛЕНИЕ

1. Введение

Проблема морфологического анализа в области обработки естественного языка заключается в том, что естественные языки имеют сложную структуру, которая может варьироваться в зависимости от контекста. Правильное определение грамматических характеристик слова, таких как часть речи, число, падеж, время и т.д., важно для многих задач обработки естественного языка, таких как автоматический перевод, распознавание речи, анализ тональности и других.

Однако традиционные методы морфологического анализа имеют ряд ограничений, такие как сложность в описании морфологических правил для каждого языка и его диалектов, необходимость постоянной ручной разметки текстов и сложность в масштабировании на большие наборы данных.

В свою очередь, нейросетевые методы морфологического анализа могут быть более точными и универсальными в сравнении с традиционными методами. Нейросетевые модели могут автоматически извлекать признаки из текста и определять грамматические характеристики слов, основываясь на больших наборах размеченных данных. Кроме того, некоторые такие модели специально могут быть масштабированы для обработки больших объемов текста.

Таким образом, разработка методов морфологического анализа на основе нейросетевых подходов является актуальной и важной задачей в области обработки естественного языка.

Необходимо отметить еще одно важное отличие между классическими и новыми методами. Традиционные методы морфологического анализа основаны на правилах и шаблонах, которые определяют морфологические характеристики слов. Традиционные методы могут быть реализованы в виде электронных морфологических словарей и морфологических анализаторов.

Морфологические словари содержат информацию о всех предусмотренных составителями словах и их характеристиках. Однако, такие словари не могут распознавать неизвестные слова или нестандартные формы слов, которые не указаны в словаре.

Морфологические анализаторы, в свою очередь, используют правила и шаблоны для определения морфологических характеристик слов. Такие анализаторы могут работать с неизвестными словами и даже с нестандартными

формами слов. Однако, создание правил для каждого языка и диалекта является достаточно трудоемкой задачей.

Нейросетевые методы морфологического анализа используют машинное обучение для определения характеристик слов. Нейросетевые модели могут использоваться для автоматического извлечения признаков из текста, определения грамматических признаков слов и лемматизации. Нейросетевые методы могут работать с нестандартными формами слов, что делает их более универсальными в сравнении с традиционными методами.

Одним из примеров нейросетевых методов является использование рекуррентных нейронных сетей (RNN) и сверточных нейронных сетей (CNN) для морфологического анализа. Рекуррентные нейронные сети позволяют учитывать контекст слова при определении его морфологических характеристик, а сверточные нейронные сети позволяют быстро и эффективно извлекать признаки из текста.

Нашей задачей будет применить упомянутые нейросетевые методы в области морфологического анализа текста, использовать размеченные данные и попробовать реализовать новый метод на основе языка программирования python.

Целью данного исследования является разработка модуля python архитектуры uniparser для русского языка на основе методов кластеризации/классификации парадигм.

Задачи работы:

1. Изучение существующих методов и инструментов для морфологического анализа русского языка;
2. Изучение теории кластеризации и классификации данных в контексте морфологического анализа;
3. Сбор и предобработка данных для обучения и тестирования модуля;
4. Разработка модели на основе выбранных методов кластеризации/классификации парадигм;
5. Реализация и тестирование модуля на выбранном наборе данных;
6. Оценка производительности и точности модуля и сравнение с другими существующими инструментами для морфологического анализа русского языка;

7. Анализ результатов исследования.

2. Теоретический обзор

2.1 Обзор существующей литературы

Методы разработки морфологических анализаторов на основе данных, размеченных нейронной сетью, являются одним из активно развивающихся направлений в области обработки естественного языка. В этой области существует множество статей, посвященных различным аспектам морфологических анализаторов на основе нейронных сетей. Мы изучем несколько уже существующих работ с использованием нейронных сетей в области морфологического анализа. Также рассмотрим исследования различных языков с различной структурой.

Кузнецов Ю. и Лесин А. (2018) написали одну из первых работ в области разработки таких анализаторов. В данной работе предложена архитектура морфологического анализатора на основе нейронной сети, которая позволяет эффективно обрабатывать слова на русском языке.

Авторы обсудили несколько недостатков традиционных подходов, основанных на правилах и словарях, при обработке сложных морфологических явлений. Затем они описали архитектуру своего морфологического анализатора на основе нейронной сети, который состоит из двунаправленной сети с долгой кратковременной памятью (LSTM) для кодирования на уровне символов, сети с прямой связью для предсказания морфологических признаков и слоя условного случайного поля (CRF) для маркировки последовательности. Авторы также обсудили процесс обучения сети, который включает в себя использование набора данных аннотированных слов и их морфологических признаков.

Юревич Э. и Кузнецов Ю. (2019) представили продолжение своей предыдущей работы над морфологическим анализатором на основе нейронной сети, включив дополнительное обучение, позволяющее добавлять новые данные для повышения точности модели.

Авторы представили процесс обучения с добавлением новых аннотированных данных в существующий набор данных и обновлением весов

модели для включения новой информации. Результаты оценки показывают, что метод поэтапного обучения повысил точность модели.

Белоногов А. и Константинова Н. (2021) представили новый подход к построению морфологического анализатора русского языка с использованием нейронных сетей. Авторы описали предложенную ими модель, в которой используется комбинация сверточных и рекуррентных нейронных сетей для выполнения кодирования на уровне символов и прогнозирования морфологических признаков.

Одной из сильных сторон этой статьи является внимание к русскому языку, который является сложным флективным языком, который создает проблемы для традиционных морфологических анализаторов, основанных на правилах. Предлагаемый подход способен обрабатывать сложную морфологию русского языка, используя возможности нейронных сетей.

Еще одной сильной стороной статьи является подробное описание предлагаемой модели и экспериментов, проведенных для оценки ее производительности.

Кессикбаева Г., Чичекли И. (2014) представили основанный на правилах морфологический анализатор казахского языка. Авторы начинают с обсуждения важности морфологического анализа в обработке естественного языка, особенно в языках со сложной морфологией, таких как казахский. Они отмечают, что, хотя существуют морфологические анализаторы для казахского языка, они имеют такие ограничения, как неполный охват данных и отсутствие гибкости. Предлагаемый анализатор состоит из набора правил анализа различных морфологических признаков казахских слов.

Фариди, А.З., и Тайерс, Ф.М. (2009) разработали морфологический анализатор с открытым исходным кодом для бенгальского языка. Авторы начинают с обсуждения важности морфологического анализа в обработке естественного языка и проблем, связанных со сложной флективной и деривационной системой бенгальского языка.

Затем авторы описали процесс разработки анализатора, который включал в себя составление исчерпывающего лексикона бенгальских слов и их

морфологических особенностей, а также создание набора правил для проведения морфологического анализа новых слов.

Дереза О. В., Каютенко Д. А., Феногенова А. С. (2016) представили сравнение десяти систем автоматического морфологического анализа: TreeTagger, Mystem, Pymorhy, Stanford POS tagger и других. Авторы сравнили производительность анализаторов с точки зрения точности и скорости обработки, а также их способность обрабатывать различные примеры.

Авторы представили четыре анализатора, выбранных для сравнительного исследования: TnT, TreeTagger, HunPos и Citar. Каждый из этих анализаторов описан с точки зрения лежащей в их основе методологии, сильных и слабых сторон.

Сегалович И. (2003) разработал модуль Mystem, который изначально был создан для поисковой системы Яндекс. Автор начал с обсуждения проблем морфологического анализа для поисковых систем, в частности проблемы неизвестных слов, которых нет в стандартных словарях. Затем он представил свой предложенный алгоритм, который использует комбинацию методов на основе словаря и правил для выполнения морфологического анализа и угадывания неизвестных слов.

Алгоритм сначала использует поиск в словаре для идентификации известных слов и их флективных форм. Затем он применяет набор правил для создания возможных форм неизвестных слов на основе их суффиксов и других лингвистических особенностей.

Коробов М. (2015) создал модуль pymorphy2, морфологический анализатор и генератор для русского и украинского языков. Автор описывает конструкцию системы, в основе которой лежит сочетание основанных на правилах и статистических методов.

Эта система состоит из нескольких компонентов, включая морфологический словарь, морфологический анализатор, лемматизатор и морфологический генератор. Морфологический словарь основан на большом корпусе русских и украинских текстов.

Сорокин А. и соавт. (2017) изучили результаты MorphoRuEval-2017, призванного стимулировать развитие технологий автоматической

морфологической обработки русского языка. Они сравнивают методы, используемые участниками для решения задачи морфологического анализа.

В этом исследовании авторы описали дизайн трека оценки, который состоял из двух заданий: морфологической маркировки и лемматизации. Задача морфологической маркировки требовала от участников присвоения морфологических меток каждому слову в заданном предложении. Оценка основывалась на корпусе текстов на русском языке, состоящем из новостных и художественных статей. В данной статье также рассматривается проблема унификации различных существующих обучающих сборников по русскому языку.

Панченко А., Константинова Н. и Лукашевич Н. (2021) представили новый подход к созданию морфологического анализатора, который может включать новую лингвистическую информацию в дополнение к стандартному вводу на уровне символов.

Сокирко А. В. (2010) описывает подход к предсказанию морфологических характеристик русских слов с использованием обширных лингвистических ресурсов. Автор представляет концепцию "быстрословаря" - системы, способной быстро и точно определять грамматические признаки слова на основе обширного словаря и грамматической информации. Автор особое внимание уделяет использованию больших лингвистических ресурсов, таких как размеченные корпусы и словари, для обучения модели и предсказания морфологических признаков слова.

Данная статья подчеркивает практическую значимость "быстрословаря" и его потенциал для широкого спектра приложений, таких как автоматическая обработка текстов, машинный перевод, информационный поиск и другие задачи, связанные с анализом и обработкой естественного языка. Сокирко А. В. представляет инновационный подход к предсказанию морфологических характеристик русских слов.

Авторы проводят эксперименты на российских и турецких наборах данных и сравнивают производительность своей модели с дополнительной лингвистической информацией и без нее. Результаты показывают, что включение данной информации повышает точность модели, особенно в тех случаях, когда

морфологический анализ зависит от контекста, выходящего за рамки самого входного слова.

2.2 Обзор методов кластеризации / классификации парадигм

Для разработки модуля Python архитектуры uniparser для русского языка на основе методов кластеризации/классификации парадигм можно использовать следующие методы исследования:

1. Кластеризация слов по формам. Можно использовать алгоритмы кластеризации, такие как k-means, DBSCAN, Agglomerative Clustering, чтобы разбить слова на группы по их формам. Например, можно сгруппировать все слова с одинаковым окончанием.
2. Классификация слов по парадигмам. Можно использовать алгоритмы машинного обучения, такие как Decision Tree, Random Forest, Naive Bayes, чтобы классифицировать слова по их парадигмам. Например, можно обучить модель, которая будет определять, к какой парадигме относится слово по его морфологическим признакам.
3. Использование статистических методов. Можно использовать статистические методы для анализа частотности различных форм слов. Например, можно использовать методы частотного анализа, такие как TF-IDF, для определения наиболее важных форм слова.
4. Анализ контекста. Можно анализировать контекст, в котором используются слова, для определения их форм и парадигм. Например, можно использовать алгоритмы NLP, такие как Named Entity Recognition, для извлечения информации о морфологических признаках слова из текста.
5. Обучение с учителем и без учителя. Можно использовать как методы обучения с учителем, такие как классификация, так и методы без учителя, такие как кластеризация, для анализа и классификации парадигм слов.

В целом, для разработки модуля Python на основе методов кластеризации/классификации парадигм можно использовать комбинацию различных методов исследования, чтобы достичь наилучших результатов.

Рассмотрим подробнее несколько существующих методов. **K-means (**Dabbura I. 2018) - это один из наиболее распространенных методов кластеризации, имеющий свою реализацию в различных библиотеках. Он основан

на разбиении множества объектов на заранее заданное количество кластеров, где каждый кластер представляет собой группу объектов, близких по своим характеристикам.

Данный алгоритм работает следующим образом:

1. Инициализация: задается количество кластеров K и случайным образом выбираются K начальных центров кластеров;
2. Присвоение объектов кластерам: каждый объект из множества данных относится к ближайшему центру кластера;
3. Пересчет центров кластеров: на основе присвоения объектов кластерам пересчитываются центры кластеров;
4. Повторение шагов 2 и 3 до тех пор, пока центры кластеров не перестанут изменяться или не будет достигнуто максимальное число итераций.

На каждой итерации алгоритма происходит пересчет расстояний между объектами и центрами кластеров, что может быть вычислительно затратно при большом количестве данных и/или большом числе кластеров.

K-means является эффективным методом кластеризации, но его результаты могут зависеть от начальных значений центров кластеров и числа кластеров.

DBSCAN или Density-Based Spatial Clustering of Applications with Noise (Lutins E. 2017) - это еще один метод кластеризации, который основывается на плотности объектов в пространстве. Он позволяет автоматически определять количество кластеров и обнаруживать выбросы (шум).

Процесс происходит таким образом:

1. Задается радиус *Eps* и минимальное количество объектов *Min_samples* для определения плотных областей;
2. Выбирается случайный необработанный объект и проверяется, есть ли в его окрестности другие объекты, находящиеся на расстоянии не более *Eps*. Если количество таких объектов больше или равно *Min_samples*, то создается новый кластер и все объекты в его окрестности добавляются в него. Если количество объектов меньше *Min_samples*, то объект помечается как шум;

3. Для каждого объекта в кластере проверяется, есть ли в его окрестности другие объекты, которые еще не были обработаны. Если есть, то эти объекты добавляются в кластер;
4. Повторяются шаги 2 и 3 до тех пор, пока все объекты не будут обработаны.

DBSCAN позволяет определять кластеры произвольной формы и устойчив к шумовым данным. Однако, при большом количестве данных и/или большой размерности пространства расчеты также могут быть вычислительно затратными. Выбор оптимальных значений *Eps* и *Min_samples* может быть нетривиальным и требует экспериментов.

Agglomerative Clustering - это метод, основанный на иерархической кластеризации. Изначально каждый объект (в нашем случае токен) считается отдельным кластером, потом ближайшие кластеры начинают объединятся до тех пор, пока не будет достигнуто определенное количество или пока все объекты не будут объединены в один большой кластер.

Расстояние между кластерами может быть определено как евклидовым расстоянием между центрами кластеров, так и минимальным расстоянием между объектами в разных кластерах. При этом получается дерево кластеров (дендрограмма), которое можно использовать для визуализации результатов.

Agglomerative Clustering имеет ряд преимуществ, таких как возможность работы с любым типом данных, возможность определения оптимального количества кластеров с помощью дендрограммы, а также возможность использования различных метрик расстояния для определения близости между кластерами.

Однако метод также имеет некоторые недостатки, такие как высокая вычислительная сложность при большом количестве объектов и необходимость определения оптимального значения параметра расстояния.

Naive Bayes (Gandhi R. 2018) - это метод машинного обучения, который используется для классификации объектов на основе вероятностных моделей. Он основан на теореме Байеса, которая позволяет вычислять вероятность того, что объект относится к определенному классу на основе его характеристик.

В методе Naive Bayes каждый объект представляется в виде набора признаков, которые описывают его свойства. Затем для каждого класса

вычисляются вероятности появления каждого из признаков и общая вероятность появления объектов этого класса. При классификации нового объекта вычисляются вероятности его принадлежности к каждому из классов на основе его признаков, и объект относится к тому классу, для которого эта вероятность максимальна.

Метод Naive Bayes называется "наивным", потому что он предполагает, что все признаки объекта независимы друг от друга, что не всегда является правдой в реальных данных. Однако этот метод все еще дает хорошие результаты во многих задачах классификации, таких как определение спама в электронной почте или распознавание рукописных цифр.

Одним из главных преимуществ метода Naive Bayes является его скорость работы и низкие требования к вычислительным ресурсам. Кроме того, он хорошо работает с большими объемами данных и может использоваться для классификации объектов с любым типом признаков.

Однако метод Naive Bayes также имеет некоторые недостатки, такие как чувствительность к выбросам в данных и неспособность учитывать сложные взаимодействия между признаками. Кроме того, он не всегда дает лучшие результаты в сравнении с такими методами, как Decision Tree или Random Forest.

Random Forest (Yiu T. 2019) - это метод машинного обучения, который используется для кластеризации объектов на основе ансамбля решающих деревьев. Он основан на идее комбинирования нескольких деревьев решений для улучшения точности и стабильности кластеризации.

В методе Random Forest каждый объект представляется в виде набора признаков, которые описывают его свойства. Затем создается несколько решающих деревьев, каждое из которых обучается на подмножестве объектов и признаков. Каждое дерево принимает решение о кластеризации объекта на основе его признаков, и объект относится к тому кластеру, который получил большинство голосов от всех деревьев.

Данный метод использует случайный выбор подмножества объектов и признаков для обучения каждого дерева. Это позволяет уменьшить вероятность переобучения и улучшить обобщающую способность модели.

Преимуществом метода Random Forest является его высокая точность и стабильность кластеризации. Он также хорошо работает с большими объемами данных и может учитывать объекты с любыми типами признаков.

Недостатки данного метода заключаются в высокой сложности модели и длительном времени обучения. Кроме того, он не способен учитывать сложные взаимодействия между признаками. В целом, метод Random Forest является мощным инструментом для кластеризации объектов, который может быть эффективно использован во многих задачах анализа данных.

В нашем исследовании мы более детально изучим данные методы и выберем наиболее подходящей нашей задаче.

2.3 Описание архитектуры uniparser и ее особенностей

Uniparser (Arkhangelskiy T. 2021) - это инструмент для автоматического разбора естественного языка на основе грамматических правил. Он использует набор грамматических правил, написанных на языке Python, для анализа входных данных и выдачи структурированной информации о частях речи, зависимостях между словами и других лингвистических аспектах. Эта библиотека является модульной и может быть использована на различных языках программирования, таких как Python, Java и C++.

Архитектура Uniparser имеет несколько особенностей:

1. Модульность: данный инструмент имеет модульную архитектуру, которая позволяет использовать различные модули для морфологического анализа и синтаксического разбора. Это означает, что пользователи могут выбрать те модули, которые наиболее подходят для конкретных задач;
2. Простота использования: Uniparser имеет простой и интуитивно понятный интерфейс, что делает его легким в использовании для пользователей с различными уровнями опыта в обработке естественного языка;
3. Лексический анализатор: он разбивает текст на отдельные слова и определяет их части речи и другие морфологические характеристики слова;
4. Синтаксический анализатор: он строит дерево зависимостей между словами и определяет их роли в предложении;

5. Семантический анализатор: он определяет значения слов и их связей на основе семантических правил.
6. Открытый исходный код: Uniparser распространяется под лицензией Apache 2.0, что позволяет свободно использовать и модифицировать его исходный код. Это также способствует развитию сообщества пользователей и разработчиков Uniparser.

Еще одной особенностью Uniparser является его способность работать с различными языками благодаря использованию универсальной модели грамматических правил. Это позволяет создавать грамматические правила для новых языков без необходимости переписывания всего инструмента.

Uniparser также предоставляет возможность настройки параметров анализа, таких как уровень детализации вывода и выбор используемых грамматических правил. Это делает его гибким инструментом для различных задач в области обработки естественного языка.

2.4 Рассмотрение примеров использования uniparser

Конкретные примеры использования Uniparser зависят от задачи и языка, на котором необходимо проводить морфологический анализ текста. Рассмотрим несколько примеров использования данной библиотеки:

- Автоматический анализ текста:
  Uniparser может быть использован для автоматического анализа текста на естественном языке. Например, он может быть использован для анализа новостных статей и выделения ключевых слов и фраз, определения смысла предложений и выявления связей между ними.
- Разработка приложений для обработки естественного языка:
  Uniparser может быть использован для разработки приложений для обработки естественного языка, таких как чат-боты или системы распознавания речи. Он может быть интегрирован в приложение, чтобы обеспечить автоматический анализ вводимого пользователем текста и создание соответствующего ответа.
- Исследовательские работы в области лингвистики:
  Uniparser может быть использован для исследовательских работ в области лингвистики, таких как анализ структуры языка или разработка новых

грамматических правил для определенного языка. Он может быть использован для автоматического анализа текстов на различных языках и создания соответствующих грамматических правил.

Конкретно в области обработки естественного языка Uniparser может использоваться для таких задач, как:

- Морфологический анализ:
  Uniparser может разбирать слова на составляющие (лемму, часть речи, падеж, число и т.д.), что может быть полезно для анализа текста на естественном языке.
- Синтаксический анализ:
  Uniparser может использоваться для построения синтаксических деревьев, которые показывают связи между словами в предложении. Это может быть полезно для автоматического извлечения информации из текста или для машинного перевода.
- Автоматический перевод:
  Uniparser может использоваться для автоматического перевода текста на другие языки, используя полученную информацию о морфологии и синтаксисе исходного текста.
- Классификация текста:
  Uniparser может использоваться для классификации текста на основе его содержания, используя информацию о морфологии и синтаксисе.

3. Подготовка данных

3.1 Особенности морфологии русского языка

Особенностями русского языка являются его сложность структуры словоизменения, флективность, наличие сложной системы чередований и другие. Также русский язык имеет сложную систему видовых противопоставлений.

Также русский язык имеет развитую систему склонения и спряжения, которая позволяет выражать различные оттенки значений слов. Например, слово "дом" может иметь различные формы в зависимости от падежа и числа: *дом, дома, дому, домов, домами, домах*.

Еще одной особенностью морфологии русского языка является наличие сложных словообразовательных процессов, таких как префиксация, суффиксация и корневые изменения. Наша задача морфологического анализа слова включает в себя определение его частей речи, склонения, спряжения и т.д., которые могут быть связаны с процессом словообразования.

Кроме того, русский язык имеет множество исключений и нестандартных форм, что усложняет его изучение и применение в речи. Например, некоторые слова имеют несколько различных форм в одном и том же падеже, а некоторые слова не склоняются вообще.

Также стоит отметить, что языковые нормы меняются, появляются различные социальные и индивидуальные “лекты”. В последнее время наблюдается упрощение склонения и спряжения в речи молодежи.

Таким образом, морфология русского языка - это одна из его главных особенностей.

Морфология русского языка отличается от морфологии других языков. Например, в английском языке меньше грамматических категорий, чем в русском, и многие формы слов не изменяются. В некоторых других языках, например, в китайском, нет грамматических падежей и окончаний слов, а смысл выражается контекстом и порядком слов в предложении.

Также морфология русского языка имеет свои особенности в сравнении с другими славянскими языками. Например, в польском языке есть больше падежей и различных форм глаголов, чем в русском. В украинском языке есть свои уникальные формы глаголов и прилагательных.

Тем не менее, морфология всех языков имеет свою сложность и требует от говорящего знания правил и умения правильно применять их в речи. Каждый язык имеет свои особенности, которые делают его уникальным и интересным для изучения.

В нашей задаче морфологического анализа языка знание его особенностей поможет в разработке правил и шаблонов для анализа словоформ, обработке сложных языковых случаев и более точном извлечении грамматических характеристик слова.

3.2 Выделение особенностей для кластеризации / классификации парадигм

Можно выделить несколько характеристик слов, которые смогут помочь более точно кластеризовать или классифицировать используемый набор данных:

1. Падеж. В русском языке шесть падежей (именительный, родительный, дательный, винительный, творительный и предложный), которые используются для обозначения грамматической функции существительного или местоимения в предложении;
2. Число. Существует два числа (единственное и множественное число), которые используются для обозначения того, относится ли существительное или местоимение к одному или нескольким лицам, животным, предметам и т. д.;
3. Род. В русском языке есть несколько родов (мужской, женский, средний, общий), которые используются для обозначения грамматического рода существительного или местоимения;
4. Спряжение глаголов. Русские глаголы изменяются по времени, виду и наклонению. Время указывает на время действия (прошедшее, настоящее или будущее), вид отражает способ представления длительности или временной структуры действия в языке, а наклонение указывает на отношение говорящего к действию;
5. Склонение прилагательных. Прилагательные согласуются в роде, числе и падеже с изменяемым существительным;
6. Краткая форма прилагательных;
7. Склонение местоимений. Местоимения также согласуются в роде, числе и падеже с замещаемым ими существительным;
8. Изменения основы. Некоторые русские слова претерпевают изменения основы при изменении формы. Например, основа слова «стул» изменяется на «стульев» в родительном падеже множественного числа. Еще один пример изменение основы глагола «брать» в «взял» в прошедшем времени;
9. Неправильные формы. Некоторые русские слова имеют неправильные формы, не соответствующие обычным правилам словоизменения. Примером супплетивных форм является слово «человек» с неправильной формой родительного падежа множественного числа («людей» вместо

«человеков»). Пример вариантных форм - это «зимой / зимою» в творительном падеже ед. числа существительных. Пример орфографически искаженной формы - это изменение основы и/или окончаний («автарафф / авторов»). Пример сокращенных форм - «тел.» для слова «телефон».

3.3 Выбор представительного набора данных для обучения и тестирования

Для разработки модуля python для русского языка необходимо выбрать представительный набор данных для обучения и тестирования.

Этот набор данных должен содержать достаточное количество примеров различных парадигм русского языка, чтобы обеспечить эффективное обучение модели. Также необходимо убедиться, что в наборе данных представлены различные типы слов (существительные, глаголы, прилагательные и т.д.), чтобы модель могла обрабатывать разнообразный текст.

Один из возможных способов выбора представительного набора данных - использование корпуса текстов на русском языке, содержащего большое количество различных слов и парадигм.

Для выбора конкретного набора данных можно использовать методы кластеризации и классификации, чтобы выделить наиболее представительные примеры различных парадигм и типов слов. Также можно использовать методы обработки естественного языка, такие как POS-теггинг и лемматизация, чтобы автоматически выделить различные формы слов и их грамматические характеристики.

Мы выделили несколько представительных наборов данных:

- **НКРЯ** (Национальный корпус русского языка): это крупнейший корпус текстов на русском языке, содержащий тексты из различных источников, включая прозу, поэзию, газеты и т.д. НКРЯ также содержит морфологически размеченные тексты, которые могут использоваться для обучения и тестирования модуля;
- **Taiga**: это открытый набор размеченных данных из разных источников. Содержит информации о морфологических характеристиках слов и синтаксических связях между ними;
- **Universal Dependencies**: это проект, который содержит размеченные корпусы текстов на различных языках, включая русский. Universal

Dependencies содержит морфологическую и синтаксическую разметку для каждого слова в тексте, что может быть полезно для разработки модуля;

- **SynTagRus**: это корпус текстов на русском языке, содержащий синтаксическую разметку для каждого предложения. SynTagRus может использоваться для обучения модуля на основе синтаксических правил;
- **OpenCorpora**: это корпус текстов на русском языке с морфологической разметкой, который содержит более 3 миллионов словоформ.

При выборе набора данных следует учитывать его размер, качество разметки и доступность для использования в разработке модуля. Лучше всего будет использовать наборы данных, которые содержат разметку для различных морфологических категорий, таких как часть речи, падеж, число и т.д., чтобы обучить модуль на широком диапазоне языковых особенностей.

3.4 Описание наборов данных

Набор морфологически размеченных данных **НКРЯ** (Национальный корпус русского языка) включает в себя большой объем текстов на русском языке, которые были размечены по частям речи, падежам, числам, временам и другим морфологическим признакам.

Набор данных НКРЯ содержит тексты из различных источников, таких как художественная литература, научные статьи, газетные и журнальные статьи, разговорная речь и т.д.

Эти данные используются для проведения лингвистических исследований, анализа текстов, создания компьютерных программ для обработки естественного языка и других целей.

База данных **Taiga** - это корпус текстов на русском языке, который содержит более 700 миллионов слов и фраз, собранных из различных источников, включая художественную литературу, публицистику, научные статьи и другие жанры.

Каждое слово в тексте имеет свой морфологический разбор, который включает информацию о его части речи, падеже, числе, роде, времени и других характеристиках. Корпус включает в себя информацию о синтаксических связях между словами в предложении, а также о стилистических особенностях текстов.

Набор данных **Universal Dependencies** (UD) - это международный проект, который предоставляет открытые и доступные данные о различных языках мира, размеченные по универсальным морфологическим признакам.

Каждое слово в тексте имеет свой морфологический разбор, который включает информацию о его части речи, падеже, числе, роде, времени и других характеристиках.

Набор данных UD включает в себя тексты на более чем 100 языках мира, в том числе на таких экзотических языках, как хакасский, карачаево-балкарский, нивхский и другие.

Данные в наборе UD используют универсальные теги для обозначения морфологических признаков, что позволяет проводить сравнительный анализ между различными языками и создавать компьютерные программы для обработки естественного языка на разных языках.

UD - это набор данных, который используется для лингвистических исследований, анализа текста, создания программ для работы на естественном языке и других задач.

SynTagRus является набором морфологических данных, которые используются для исследования синтаксиса и грамматики русского языка. Данный набор данных основан на Корпусе русского языка. В него входят различные тексты, такие как проза, поэзия, научные тексты и другие, написанные на русском и других языках.

Каждому слову в предложении приписаны морфологические тэги и грамматические атрибуты. Такая разметка позволяет исследователям изучать различные языковые явления, такие как грамматические конструкции, зависимости между словами, синтаксические структуры предложений и многое другое.

Набор данных SynTagRus содержит более 500 000 словоформ, которые разделены на предложения и размечены с использованием морфологических тэгов. Каждое слово в предложении имеет свой уникальный идентификатор, морфологическую информацию (такую как падеж, род, число, время и т.д.) и связи с другими словами в предложении.

Этот набор данных позволяет проводить различные эксперименты, тренировать модели и создавать приложения, основанные на анализе и синтаксической обработке текстов на русском языке.

Набор данных **OpenCorpora** - это корпус текстов на русском языке, размеченных по морфологическим признакам.

Каждое слово в тексте имеет свой морфологический разбор. Кроме того, данные в наборе OpenCorpora включают информацию о грамматических формах слов, например, о склонении глаголов и прилагательных, и включают уникальные теги для обозначения морфологических признаков, что позволяет проводить более точный анализ текстов на русском языке.

Набор данных OpenCorpora включает в себя большое количество текстов различных жанров, таких как художественная литература, научные статьи, газетные материалы и другие.

3.5 Предобработка данных для подготовки их к вводу в нейросеть

Перед обучением модели необходимо провести предобработку данных, чтобы подготовить их к вводу в нейросеть. Важными этапами предобработки данных являются:

1. Токенизация - разбиение текста на отдельные слова или токены. Это необходимо для того, чтобы нейросеть могла обрабатывать каждое слово отдельно;
2. Лемматизация - приведение слов к их базовым формам (леммам). Это позволяет сократить количество уникальных слов в наборе данных и уменьшить размерность входных данных для нейросети;
3. POS-теггинг - определение грамматических характеристик каждого слова (часть речи, падеж, число и т.д.). Эта информация может быть полезна для определения парадигм и правил склонения/спряжения;
4. Удаление стоп-слов - удаление слов, не несущих никакой смысловой нагрузки (например, предлогов, союзов и т.д.). Это позволяет уменьшить размерность входных данных и ускорить обучение модели;
5. Векторизация - преобразование слов в числовые векторы. Для этого можно использовать различные методы, такие как Bag-of-Words, TF-IDF или

Word2Vec. Это позволяет представить слова в виде числовых значений, которые могут быть обработаны нейросетью.

После проведения предобработки данных и получения числовых векторов для каждого слова, можно приступать к обучению модели на выбранном наборе данных. В нашем исследовании мы используем уже размеченные корпуса текстов, содержащие всю необходимую информацию о каждом из токенов. Комбинация различных датасетов, использование текстов с различными тематиками, размер обучающих данных - все это может повлиять на результаты работы нейронной сети.

4. Разработка модуля на основе методов кластеризации парадигм

4.1 Выбор подходящей модели предобработки

Открытые датасеты GSD, SynTagRus, Taiga представлены в формате conllu (Conference on Computational Natural Language Learning), содержащий в себе всю необходимую информацию о предложении и словах, морфологии, синтаксисе и семантике.

В представленных данных используется универсальный набор тегов для обозначения различных частей речи:

- ADJ – имя прилагательное
- ADP – предлог
- ADV – наречие
- AUX – вспомогательный глагол
- CCONJ – соединительный союз
- DET – определяющее слово
- INTJ – междометие
- NOUN – имя существительное
- NUM – числительное
- PART – частица
- PRON – местоимение
- PROPN – имя собственное
- PUNCT – пунктуация

- SCONJ – подчинительный союз
- SYM – символ
- VERB – глагол
- X – иностранные слова и т. д.

Данные, выбранные нами для задачи морфологического анализа текста, представлены в разных объемах, но в схожих форматах. Каждый датасет включает в себя следующую информацию: id предложения, предложение, id слова, словоформу, лемму, часть речи, id главного слова (от которого зависит текущий токен), отношение между главным и зависимым в формате синтаксических отношений UD, список вторичных зависимостей и другие аннотации.

У каждой части речи также есть свои особенности, перечисленные для каждого слова: одушевленность / неодушевленность, совершенный / несовершенный вид, падеж (именительный, родительный, дательный, винительный, творительный, партитивный, местный, звательный), степень (положительная, сравнительная), иностранное / не иностранное слово, род (мужской, женский, средний), единственное / множественное число, лицо (первое, второе, третье), отрицательная полярность, время (прошедшее, настоящее, будущее), краткость, форма глагола (начальная, инфинитив, причастие, деепричастие), залог (активный, пассивный).

Таблица 1. Число токенов в используемых датасетах

Датасет	**Обучающие данные (кол-во токенов)**	**Валидационные данные (кол-во токенов)**	**Тестовые данные (кол-во токенов)**
GSD	74900	11710	11385
SynTagRus	1206300	153590	157990
Taiga	176630	10095	10275

В нашем эксперименте будут использованы основные характеристики слова, а именно начальная форма, часть речи и ее особенности. Для каждой словоформы будет представлен свой набор признаков.

Например, для слова “наследство” будет список “наследство,NOUN,Inan,Acc,Neut,Sing”, указывающий на то, что слово является

неодушевленным существительным среднего рода, стоящим в винительном падеже в единственном числе. В случаях с пунктуацией характеристика будет описана в виде “PUNCT,None”.

Рисунок 1. Пример полученной базы данных размеченных слов

sent_id	text	id	form	lemma	upos	xpos	feats	head	deprel	deps	misc
train-s1	Во время битвы между силами Магнето	1	Во	во	ADP	IN		2	case		
train-s1	Во время битвы между силами Магнето	2	время	время	NOUN	NN	{'Animacy': 'Inan', 'Case': 'Acc', 'Gender': 'Neut', 'Number': 'Sing'}	18	obl		
train-s1	Во время битвы между силами Магнето	3	битвы	битва	NOUN	NN	{'Animacy': 'Inan', 'Case': 'Gen', 'Gender': 'Fem', 'Number': 'Sing'}	2	nmod		
train-s1	Во время битвы между силами Магнето	4	между	между	ADP	IN		5	case		
train-s1	Во время битвы между силами Магнето	5	силами	сила	NOUN	NN	{'Animacy': 'Inan', 'Case': 'Ins', 'Gender': 'Fem', 'Number': 'Plur'}	3	nmod		
train-s1	Во время битвы между силами Магнето	6	Магнето	Магнето	PROPN	NNP	{'Animacy': 'Anim', 'Case': 'Gen', 'Gender': 'Masc', 'Number': 'Sing'}	5	nmod		
train-s1	Во время битвы между силами Магнето	7	и	и	CCONJ	CC		8	cc		
train-s1	Во время битвы между силами Магнето	8	героями	герой	NOUN	NN	{'Animacy': 'Anim', 'Case': 'Ins', 'Gender': 'Masc', 'Number': 'Plur'}	5	conj		{'SpaceAfter': 'No'}
train-s1	Во время битвы между силами Магнето	9	,	,	PUNCT	,		11	punct		
train-s1	Во время битвы между силами Магнето	10	кто	кто	PRON	WP	{'Animacy': 'Anim', 'Case': 'Nom', 'Gender': 'Masc', 'Number': 'Sing'}	11	nsubj		
train-s1	Во время битвы между силами Магнето	11	восстановил	восстановить	VERB	VBC	{'Aspect': 'Perf', 'Gender': 'Masc', 'Mood': 'Ind', 'Number': 'Sing', 'Ten	8	acl:relcl		
train-s1	Во время битвы между силами Магнето	12	свои	свой	DET	PRP$	{'Animacy': 'Inan', 'Case': 'Acc', 'Number': 'Plur'}	13	det		
train-s1	Во время битвы между силами Магнето	13	воспоминания	воспоминание	NOUN	NN	{'Animacy': 'Inan', 'Case': 'Acc', 'Gender': 'Neut', 'Number': 'Plur'}	11	obj		{'SpaceAfter': 'No'}

Рисунок 2. Пример подготовленной к обучению базы данных размеченных слов

form	data
Начальный	начальный,ADJ,Nom,Pos,Masc,Sing
ролик	ролик,NOUN,Inan,Nom,Masc,Sing
,	,,PUNCT,None
или	или,CCONJ,None
опенинг	опенинг,NOUN,Inan,Nom,Masc,Sing
(	(,PUNCT,None
от	от,ADP,None
,	,,PUNCT,None
сокр.	сокращенно,ADV,Pos

4.2 Анализ примененных методов кластеризации

После приведения данных в единый формат, мы приступаем к задаче кластеризации слов. В нашем исследовании данная процедура поможет в обучении нейронной сети, так как она позволит сократить количество уникальных слов в обучающем корпусе и преобразовать их в числовые значения. Это позволит уменьшить размерность входных данных для нейронной сети и ускорить ее обучение. Кроме того, кластеризация может помочь выделить смысловые группы слов, что может улучшить качество предсказаний нейронной сети.

4.2.1 Кластеризация с помощью K-means

Мы применили метод K-means на наших данных. Одной из главных сложностей является выбор количества необходимых кластеров. В рамках данного эксперимента мы установили количество, равное одной десятой от количества токенов в датасете (например, для обучающего датасета GSD данное число равно

7490). Полученные кластеры довольно разнообразны по своей длине и наполнению. Количество токенов в одном кластере может достигать 19000 единиц, в то время как другой кластер содержит всего 2 единицы. При этом средняя длина кластера составляет 11 единиц.

Рисунок 3. Пример образованных кластеров с помощью метода K-means

Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
пустое	права	кровь	Сериал	начало	использован
пусть	правах	крови	сериалах	началом	использована
Капустин	Справа	акров	сериале	означало	использовав
Пустошь	справа	сокровищах	сериала	Начало	использованы
Пустоте	Права		Сериал		
устья			минисериала		
			сериала		
			сериале		

В дальнейшем обучении данным будет присваиваться номер кластера, что должно как ускорить обучение модели нейронной сети, так и улучшить предсказание для новых входных данных.

Кластеризация большого объема данных с помощью метода K-means занимает много времени. Обучающий датасет Taiga, состоящий из 176630 токенов, был кластеризирован с помощью библиотеки sklearn (установленное количество кластеров - 3532, 1/50 от количества слов). Для реализации данного метода потребовалось 3 часа 15 минут. Аналогичным образом был кластеризирован валидационный датасет Taiga, в итоге получено 1169 различных сгруппированных списков слов.

Метод K-means объединяет переданные ему формы слов в парадигмы или группы со схожим написанием или строением слова. Единичные и уникальные вхождения токенов могут привести к тому, что кластер будет состоять из одного элемента.

Рисунок 4. Пример образованных кластеров с помощью метода K-means

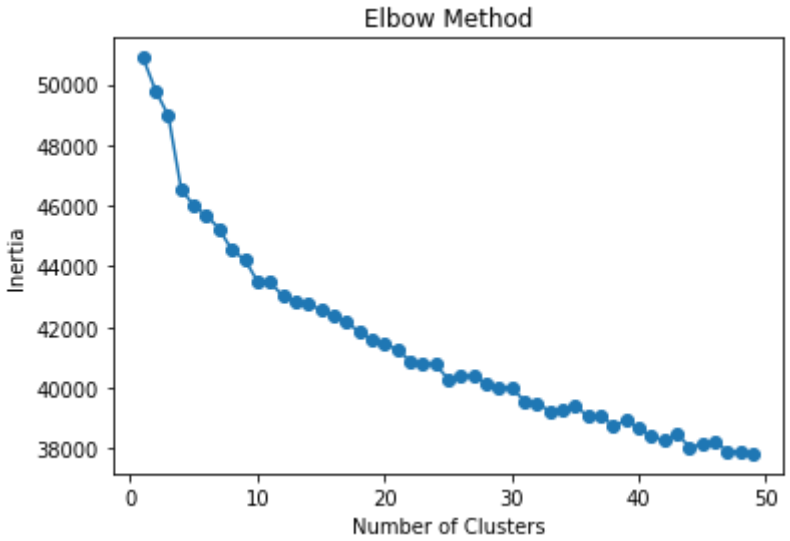
Cluster 1164	Cluster 1165	Cluster 1166	Cluster 1167	Cluster 1168	Cluster 1169
математике	Конгресс	Светлановским	заседанию	пресвитерианской	освобождённой
кинематика	Конгресса		заседания		освобождённая

Выбор оптимального количества кластеров для данного метода может быть нетривиальной задачей и зависит от специфики данных, а также от конкретных требований задачи. Однако существуют некоторые методы и подходы, которые могут помочь в выборе оптимального количества кластеров.

Один из таких подходов - использование метода "локтя" (Elbow method). Этот метод основан на наблюдении за изменением суммы квадратов расстояний внутри кластеров (инерции) при различном количестве кластеров. Обычно с увеличением числа кластеров инерция будет уменьшаться, но с некоторого момента изменение станет менее значительным (начнется "сгиб"), что может указывать на оптимальное количество кластеров.

В нашем эксперименте мы создаем список числа кластеров *num_clusters*, для каждого значения которого мы обучаем модель *K-means* и сохраняем значение инерции (*inertia_*). Затем мы визуализируем изменение инерции в зависимости от числа кластеров и ищем точку, где изменение становится менее значительным.

Рисунок 5. Пример изменения значения инерции при 1-50 кластерах

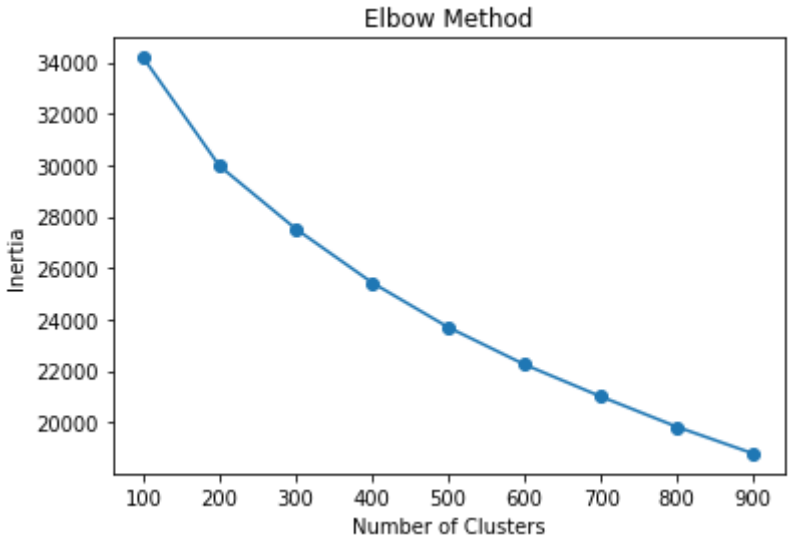


После визуальной оценки графика мы можем выбрать оптимальное количество кластеров на основе точки "сгиба" на графике инерции. Однако решение о выборе оптимального числа кластеров принимается с учетом дополнительной информации о данных и целях нашей задачи.

Применим еще раз метод “локтя” с большим количеством кластеров. В данном случае метод может не дать четкого результата, так как при увеличении

числа кластеров сумма квадратов расстояний между точками и центрами кластеров будет продолжать уменьшаться, но это не будет означать, что полученное разбиение на кластеры является оптимальным. Поэтому важно выбирать количество кластеров, которое дает наибольшее уменьшение суммы квадратов расстояний, но при этом сохраняет интерпретируемость и практическую значимость полученных кластеров.

Рисунок 6. Пример изменения значения инерции при 100-1000 кластерах



При кластеризации используемых нами данных мы использовали различное количество кластеров. Для датасета GSD необходимое число было выбрано по принципу одной десятой от количества токенов. Метод K-means показал такие результаты:

Таблица 2. Статистика кластеризованных данных GSD с помощью метода K-means

Датасет	**Количество токенов**	**Количество кластеров**	**Количество токенов в самом большом кластере**	**Количество токенов в самом маленьком кластере**
GSD train	74900	7490	19137	1
GSD dev	11709	1170	3520	1

В то же самое время для датасета Taiga было выбрано количество кластеров, равных одной пятидесятой от количества токенов в датасете. Таким образом

данному набору морфологически размеченных данных соответствует такая статистика:

Таблица 3. Статистика кластеризованных данных Taiga с помощью метода K-means

Датасет	**Количество токенов**	**Количество кластеров**	**Количество токенов в самом большом кластере**	**Количество токенов в самом маленьком кластере**
Taiga train	176630	3531	49172	1
Taiga dev	10095	200	2988	1

Кластеризация данных с помощью метода k-means достаточно удобна и эффективна в задаче кластеризации слов и символов. При правильном выборе параметров алгоритма, таких как количество кластеров и расстояние между объектами, можно добиться максимально точного разбиения токенов на группы. Кроме того, k-means является одним из наиболее распространенных методов кластеризации, который имеет многочисленные реализации в различных библиотеках и программных пакетах, что делает его доступным для широкого круга пользователей и удобным в использовании. Однако, как и любой другой метод кластеризации, k-means не всегда может давать оптимальные результаты и может требовать дополнительной, более детальной и долгой настройки для достижения поставленных целей.

4.2.2 Кластеризация с помощью DBSCAN

Мы применили метод DBSCAN на наших данных и получили следующие результаты. Для обучающего датасета GSD было получено 6359 различных кластеров, для валидационного датасета GSD было получено 982 кластера. В результате для применения метода DBSCAN библиотеки sklearn потребовалось 2 часа 55 минут, что на 20 минут быстрее результатов обработки с помощью метода K-means той же библиотеки.

Для обучающего датасета Taiga было получено 3598 различных кластеров, для валидационного датасета Taiga было получено 934 кластера. Окончательные итоги применения данного метода на используемых нами наборах морфологически размеченных данных представлены в таблице:

Таблица 4. Статистика кластеризованных данных GSD и Taiga с помощью метода DBSCAN

Датасет	**Количество токенов**	**Количество кластеров**	**Количество токенов в самом большом кластере**	**Количество токенов в самом маленьком кластере**
GSD train	74900	3659	20294	2
GSD dev	11709	982	4539	2
Taiga train	176630	3598	47349	5
Taiga dev	10095	934	3458	2

Полученные кластеры представляют собой наборы максимально схожих между собой токенов, в основном один кластер состоит из одной и той же словоформы одной леммы, как можно увидеть на примере:

Рисунок 7. Примеры образованных кластеров с помощью метода DBSCAN

Cluster 1	Cluster 60	Cluster 426	Cluster 1187	Cluster 1169	Cluster 1200	Cluster 1214
!	Машину	народу	народа	народов	тарифы	НаЧАлА
!	Машину	народу	народа	народов	тарифы	начала
.	машину	народу	народа	народов	тарифы	начала
.	машину	народу	народа	народов	тарифы	начала
,	машину	народу	народа	народов	тарифы	начала
,	машину	народу	народа	народов		начала

Все уникальные словоформы, редкие и иностранные слова были объединены в один кластер, который включил в себя почти треть от общего количества входных данных:

Рисунок 8. Пример самого крупного кластера, образованного с помощью метода DBSCAN

Cluster -1					
международного	father	Водах	Атырау	Сооружение	ячейку
международное	fibrōsa	Водителю	Аутентичная	Соц	👰👶
международном	foramĭna	Военторге	Аннунциатой	Сомов	🍿)))
международных	game	Вожделенней	АмурГостиница	Соляном	😉✌️
народе	grow	Возрадуйтесь	Амп.	Сомали	ёмкостью
народная	gt	Воино-Ясенецкий	Амели	Сомов	ёй
народного	he	Волан-де-Морт	Аменадилем	Сообщений	ёё

Такой результат может означать, что алгоритм слишком сильно уделяет внимание схожести слов, игнорируя другие признаки, такие как контекст или частота употребления. В этом случае можно изменить параметры DBSCAN, чтобы

учитывать другие признаки. Таким образом установленные нами параметры кластеризации *eps=0.3* (параметр *epsilon* определяет радиус окружности вокруг каждой точки, в пределах которой находятся ее соседи; этот параметр влияет на размер кластеров: чем больше eps, тем больше точек попадает в кластеры) и минимальное количество примеров равное 5 не привело нас к ожидаемому разбиению датасета на парадигмы.

При обучении модели DBSCAN с параметрами *eps=0.1* и *min_samples=1* возможны кластеры, состоящие из одного токена. Засчет такого изменения количество полученных кластеров может вырасти в несколько раз. Для сравнения, у валидационного датасета GSD (первый набор параметров: *eps=0.3, min_samples=2*; второй набор параметров: *eps=0.1, min_samples=1*) количество кластеров выросло с 982 до 5562, размер самого большого кластера уменьшился с 4539 до 2850 токенов. У обучающего датасета GSD (параметры идентичны параметрам валидационного датасета) количество кластеров выросло с 3659 до 24434, размер самого большого кластера уменьшился с 20294 до 18291 токенов.

Для валидационного датасета Taiga результаты схожи. Мы изменили набор параметров *eps=0.3* и *min_samples=2* на *eps=0.1* и *min_samples=1*. Таким образом количество кластеров увеличилось с 934 до 4558, а количество токенов в самом большом кластере уменьшился с 3458 до 2090. В случае с обучающим датасетом результаты двух различных обучений метода DBSCAN имеют кардинальные различия. При параметрах *eps=0.3* и *min_samples=5* количество кластеров и длина самого большого кластера равнялись 3598 и 47349 соответственно. При запуске обучения с *eps=0.1* и *min_samples=1* количество кластеров и длина самого большого кластера стали 33843 и 47329 соответственно.

Пример полученных групп при условии, что кластер может содержать от 1 токена:

Рисунок 9. Примеры образованных кластеров с помощью метода DBSCAN (min_samples=1)

Cluster 6	Cluster 60	Cluster 94	Cluster 183	Cluster 185	Cluster 245	Cluster 400
у	генерала	США	--	на	≠	Гуиннер
у	генерала	США	--	на	人	
у	генерала	США	--	на	国	
э	генерала	США	--	на	外	
э		США	--	на	奏	
я		США	--	на	抄	

Какие еще параметры можно попробовать настроить для достижения наибольшей точности разбиения на кластеры? Метод DBSCAN библиотеки sklearn предлагает такой выбор:

- *metric*: определяет метрику расстояния между точками. По умолчанию используется евклидова метрика, но можно выбрать другую метрику из библиотеки *scipy.spatial.distance*;
- *algorithm*: определяет алгоритм поиска соседей точки. Обычно используется kd-дерево, но можно выбрать brute-force алгоритм, который перебирает все точки;
- *leaf_size*: определяет размер листа в kd-дереве. Чем больше лист, тем быстрее работает алгоритм, но может возрасти память, занимаемая деревом;
- *p*: определяет параметр манхэттенской метрики или метрики Минковского. По умолчанию *p=2*, что соответствует евклидовой метрике;
- *metric_params*: дополнительные параметры для метрики расстояния;
- *n_jobs*: определяет количество ядер процессора, которые будут использоваться для вычислений. Если *n_jobs=-1*, то используются все доступные ядра.

Подбор данных параметров поможет достичь оптимального результата в нашей задаче разбиения большого набора данных на парадигмы.

4.2.3 Кластеризация с помощью Agglomerative Clustering

Метод Agglomerative Clustering библиотеки sklearn также был выбран для кластеризации данных GSD и Taiga. Отличие данного способа от DBSCAN заключается в том, что число кластеров задается на этапе обучения модели. Этот

алгоритм начинает с каждой точки в отдельном кластере и последовательно объединяет близкие кластеры до тех пор, пока не будет достигнуто заданное количество кластеров.

В отличие от этого, DBSCAN определяет количество кластеров на основе структуры данных. Алгоритм ищет плотные области в данных и считает, что каждая такая область является кластером.

Для используемых нами датасетов было выбрано фиксированное количество кластеров: для обучающих наборов данных это 7000 кластеров, для валидационных - 1000. В среднем для применения данного метода потребовалось 3 часа 28 минут. Полученные результаты представлены в таблице:

Таблица 5. Статистика кластеризованных данных GSD и Taiga с помощью метода Agglomerative Clustering

Датасет	**Количество токенов**	**Количество кластеров**	**Количество токенов в самом большом кластере**	**Количество токенов в самом маленьком кластере**
GSD train	74900	7000	18579	1
GSD dev	11709	1000	7672	1
Taiga train	176630	7000	42381	1
Taiga dev	10095	1000	5643	1

Аналогично работе метода DBSCAN метод Agglomerative Clustering объединяет в один кластер уникальные токены, которые сложно выделить в отдельные кластеры. Также в данный кластер попадают самые частотные слова (предлоги, союзы и другие) и пунктуация. Пример набора токенов, объединенных по данному принципу:

Рисунок 10. Примеры образованных кластеров с помощью метода Agglomerative Clustering

Cluster 7	Cluster 149	Cluster 203	Cluster 400	Cluster 567	Cluster 794	Cluster 1000
у	предастся	Анзерский	анорексией	Открытый	Меркатор	знать
я	создаётся	Белозерский	коррупцией	открыты	перебежчика	познавать
феях	удастся	Зангезурский		открытые	переборка	сознавать
ч	удаётся			открытый		узнать
экшн				открытой		Узнать
язв				полуоткрытыми		

Для данного алгоритма кластеризации также существует определенный набор параметров, который может помочь достичь необходимого нам результата:

- *n_clusters*: количество кластеров, которые нужно найти. По умолчанию равно 2;
- *affinity*: метрика расстояния между объектами. Может быть выбрана из предопределенных (например, "euclidean", "manhattan" или "cosine") или задана пользователем. По умолчанию используется "euclidean";
- *linkage*: метод связывания кластеров при объединении. Может использоваться один из известных ("ward", "complete", "average", "single"). По умолчанию используется "ward";
- *distance_threshold*: порог расстояния для объединения кластеров. Если не задан, то используется “n_clusters”;
- *compute_full_tree*: флаг, указывающий, нужно ли вычислять полное дерево связей между кластерами;
- *memory*: объект для кеширования промежуточных результатов;
- *connectivity*: матрица связей между объектами. Может быть использована для ограничения связей между объектами и ускорения вычислений.

Для данного метода можно построить дендрограмму, графическое представление иерархической структуры кластеров в данных. На дендрограмме расстояние между кластерами отображается по вертикальной оси, а объекты или кластеры объединения отображаются по горизонтальной оси. Такой способ визуализации может помочь определить оптимальное количество кластеров, выбрав точку на вертикальной оси, где расстояние между группами начинает резко увеличиваться. Это позволит нам лучше понимать, какие объекты или группы объектов находятся ближе друг к другу и как они связаны.

Данный метод кластеризации удобен, когда данные имеют иерархическую структуру и нужно получить не только число кластеров, но и их взаимосвязь. Однако Agglomerative clustering требует большей вычислительной мощности и времени на обработку большого объема данных.

4.3 Обучение модели на выбранном наборе данных

Для наших задач были выбраны несколько разных моделей нейронных сетей, применимых в области обработки большого объема текста. Каждая из этих

нейросетей была обучена на заранее кластеризованных нами данных. Рассмотрим их поподробнее.

4.3.1 T5-small

T5-small - это модель нейронной сети, которая представляет собой одну из предобученных моделей из семейства T5 (Text-To-Text Transfer Transformer). T5-small является наименьшей по размеру и наименее сложной версией модели T5, но все равно обладает значительной мощностью и способностью генерировать текст на основе заданных входных данных.

Архитектура модели T5-small основана на трансформере и состоит из кодировщика и декодировщика. Модель использует множество слоев трансформера для обработки последовательности данных и улавливания контекстуальных зависимостей в тексте. Она обучается на огромных объемах текстовых данных с использованием метода self-supervised learning, что позволяет модели изучить языковые закономерности и создавать связный и грамматически корректный текст.

Одной из ключевых особенностей модели T5-small является ее универсальность в обработке текста. Она может использоваться для широкого спектра задач, таких как машинный перевод, ответы на вопросы, генерация текста, автозаполнение и многое другое. В нашем случае модель используется для генерации морфологических признаков нового слова. Данная модель может быть дообучена на конкретных задачах с помощью дополнительных данных и тонкой настройки, что позволяет ей достичь более точных результатов для специфических задач.

Однако следует отметить, что T5-small - это относительно маленькая модель по сравнению с более крупными вариантами T5, такими как T5-base или T5-large. Это означает, что она может иметь ограниченные возможности в генерации текста с высоким качеством в сравнении с более крупными моделями. Тем не менее данная модель быстрее обучается и требует меньшей вычислительной мощности, что позволяет запускать обучение большое количество раз и пробовать разные вариации обучающих данных.

При обучении наших данных большее количество шагов (steps) показывало большее уменьшение функции потерь (loss). Шаг обучения в нейронной сети - это

один проход через всю тренировочную выборку. Каждый шаг обучения состоит из нескольких этапов: подачи входных данных в сеть, вычисления значений на каждом слое, расчета ошибки и корректировки весов, чтобы минимизировать потери на следующем шаге. Шаги обучения используются для улучшения точности модели и ее способности обобщать полученные знания на новые данные.

Полученные промежуточные веса модели *t5-small* сохранялись на этапах 1000, 10000, 50000 и 100000 шагов (для разных датасетов, используемых в обучении, данные этапы различаются). На примере обучения датасета SynTagRus в среднем функция потерь на обучающих данных уменьшалась с 3,247 до 0,178, на валидационных данных с 4.164 до 2.666. Цель обучения нейронной сети - минимизировать данный показатель. На каждом шаге обучения модель генерирует предсказание, а затем сравнивает его с ожидаемым результатом (например, правильные признаки слова в нашей задаче). Loss определяется как расстояние между этими двумя значениями. Чем ниже значение loss, тем более точной считается модель.

Мы провели несколько экспериментов по запуску данной модели на обучение. Лучшие показатели функции потерь достигались при наибольшем количестве шагов обучения. Однако на определенном этапе обучения loss уже не уменьшается, а остается почти тем же. Это может означать, что модель достигла своего предела в определении паттернов в данных, и дальнейшее обучение не приведет к улучшению результатов. В таком случае, мы попробовали изменить гиперпараметры. Для решения данной проблемы также можно использовать более сложные модели для достижения лучших результатов.

4.3.2 RuPrompts

RuPrompts - это модель нейронной сети, разработанная для генерации текста на русском языке. Она основана на архитектуре GPT (Generative Pre-trained Transformer), которая является одной из самых успешных архитектур для задач генерации текста.

Архитектура модели ruprompts состоит из множества слоев трансформера, которые позволяют модели эффективно обрабатывать последовательности данных. Модель также обучается на больших объемах текстовых данных.

Уникальностью модели ruprompts является наличие промптов (подсказок), которые помогают модели генерировать текст с заданной структурой и стилем. Промпты могут быть в виде инструкций, заголовков или частично заполненных предложений, что позволяет пользователю контролировать выходной текст и указывать желаемые характеристики.

Модель ruprompts имеет большой объем параметров, что позволяет ей обучаться на разнообразных данных и генерировать высококачественный текст с соблюдением языковых правил и стилей. Она может быть использована для различных задач, таких как генерация текста, ответы на вопросы, автозаполнение текста и многое другое. Однако следует отметить, что конкретная реализация модели ruprompts может варьироваться в зависимости от версии и специфических деталей реализации.

В процессе обучения каждые 50 шагов происходило оценивание модели, происходило изменение потери на обучающем и валидационном датасетах. Например, при обучении кластеризованных данных GSD loss уменьшался с 0.911 и 0.887 до 0.146 и 0.104 на датастах для обучения и валидации. Обучение такой модели с выбранными параметрами (размер батча обучающего датасета равный 2, валидационного датасета равный 2, изменение функции потерь каждые 50 шагов, коэффициент обучения равный 0.1) при заданном максимальном количестве шагов в 100000 заняло около 84 часов.

При запуске экспериментов с моделью RuPrompts мы также столкнулись с проблемой выхода loss на некое плато, нейронная сеть перестает обучаться и дальнейшее обучение не даст значительного прироста в точности предсказаний. Еще одной проблемой, с которой мы столкнулись при обучении и дальнейшем тестировании результатов обучения, является недостаток данных. Применяемые нами методы кластеризации должны были частично решить эту задачу, однако количество токенов в кластере и общее количество образованных кластеров также влияет на точность предсказаний обученной модели.

4.4 Оценка производительности модуля на тестовом наборе данных

Для того, чтобы оценить результаты различных запусков моделей нейронных сетей на обучение, мы выбрали несколько распространенных методов, которые сравнивают истинные значения и предсказанные нейронной сетью.

Формат тестовых данных представлен на рисунке:

Рисунок 11. Примеры тестовых данных Taiga

Form	Data
Ещё	еще,ADV,Pos
зимой	зима,NOUN,Inan,Ins,Fem,Sing
в	в,ADP,None
армиях	армия,NOUN,Inan,Loc,Fem,Plur
ДНР	ДНР,PROPN,Yes,Geo
и	и,CCONJ,None
ЛНР	ЛНР,PROPN,Yes,Geo

Различные методы сравнения схожести строк имеют свои подходы и особенности. Приведем краткое описание каждого из них:

1. Метод Левенштейна (Levenshtein similarity):

   Также известный как редакционное расстояние, измеряет минимальное количество операций (вставка, удаление и замена символов), необходимых для преобразования одной строки в другую. Вычисление расстояния Левенштейна между двумя строками позволяет оценить их схожесть. Процент схожести определяется путем вычисления отношения расстояния Левенштейна к максимальной длине строк. Большее значение процента схожести указывает на более похожие строки.

2. SequenceMatcher из модуля *difflib*:

   SequenceMatcher предоставляет функциональность для сравнения последовательностей, включая строки. Он вычисляет "отношение сходства" между двумя последовательностями путем сравнения их элементов и вычисления доли совпадающих элементов.

3. Косинусное сходство (Cosine similarity):

   Косинусное сходство является метрикой, которая измеряет косинус угла между двумя векторами в многомерном пространстве. В случае сравнения строк, каждая строка представляется вектором на основе встречаемости или весовых значений слов в ней (в нашем случае с помощью метода TF-IDF). Затем вычисляется косинусное сходство между векторами строк.

4. Коэффициент Жаккара (Jaccard similarity):

Данный коэффициент используется для измерения схожести между двумя множествами. В случае сравнения строк, каждая строка рассматривается как множество символов. Коэффициент Жаккара вычисляется как отношение размера пересечения множеств к их объединению.

5. Метод Jaro-Winkler:

Представляет собой алгоритм для измерения схожести между двумя строками. Он основан на расчете различных метрик, таких как длина общего префикса, длина общей последовательности и различные весовые коэффициенты. Алгоритм начинает с определения длины наибольшего общего префикса между двумя строками. Коэффициент сходства Jaro вычисляется путем комбинирования различных метрик. Метод Jaro-Winkler включает дополнительную модификацию, называемую "бонусом Winklera". Этот бонус увеличивает коэффициент сходства для строк, которые имеют близкое совпадение префиксов.

Мы протестировали полученные нами нейронные сети, получив для каждого слова из тестовой выборки предсказание от модели. Формат тестовых данных с добавленными предсказаниями представлен на рисунке:

Рисунок 12. Примеры тестовых данных с предсказанием модели

Form	Data	Predicted data
танков	NOUN,Inan,Gen,Masc,Plur	NOUN,Inan,Gen,Fem
и	CCONJ,None	CCONJ,None
тысяча	NOUN,Inan,Nom,Fem,Sing	NOUN,Inan,Nom,Fem
боевых	ADJ,Gen,Pos,Plur	ADJ,Nom,Pos,
бронированных	ADJ,Gen,Pos,Plur	ADJ,Nom,Pos,
машин	NOUN,Inan,Gen,Fem,Plur	NOUN,Inan,Acc,Masc
.	PUNCT,None	PUNCT,None

Для того, чтобы получить более точные предсказания, не зависящие от недоработанного предсказания нейронной сетью начальной формы слова, мы будем оценивать тестовые данные на основе грамматических признаков слова. Для каждых полученных результирующих весов модели нейронной сети мы также

получили процент совпадения каждого из отдельных признаков и потом вычислили среднее для каждой из пар истинных и предсказанных данных.

В результате обучения с помощью модели T5 на кластеризованных с помощью K-means данных датасета GSD получились данные метрики:

Таблица 6. Полученный процент точности датасета GSD на модели T5 (класт. K-means)

Название метода тестирования	**Средний процент точности**	**Средний процент точности при разбиении на признаки**
Levenshtein distance	22,15%	26,85%
Sequence Matcher	26,42%	30,21%
Cosine similarity	13,42%	-
Jaccard similarity	23,02%	27,54%
Jaro-winkler	38,74%	39,32%

Мы также получили метрики для той же модели и метода кластеризации, но для датасета Taiga:

Таблица 7. Полученный процент точности датасета Taiga на модели T5 (класт. K-means)

Название метода тестирования	**Средний процент точности**	**Средний процент точности при разбиении на признаки**
Levenshtein distance	67,03%	71,8%
Sequence Matcher	72,91%	79,39%
Cosine similarity	59,45%	-
Jaccard similarity	68,11%	72,72%
Jaro-winkler	84,41%	81,64%

Рассмотрим метрики для модели T5, датасета Taiga, кластеризованного с помощью метода Agglomerative Clustering:

Таблица 8. Полученный процент точности датасета Taiga на модели T5 (класт. AC)

Название метода тестирования	**Средний процент точности**	**Средний процент точности при разбиении на признаки**
Levenshtein distance	68,17%	72,56%
Sequence Matcher	73,93%	79,95%
Cosine similarity	60,66%	-
Jaccard similarity	69,24%	73,5%
Jaro-winkler	85,15%	81,93%

Для второй модели нейронной сети мы также получили свои результаты. Для датасета GSD, кластеризованного с помощью метода K-means, получены такие метрики:

Таблица 9. Полученный процент точности датасета GSD на модели RuPrompts (класт. K-means)

Название метода тестирования	**Средний процент точности**	**Средний процент точности при разбиении на признаки**
Levenshtein distance	52,42%	51,36%
Sequence Matcher	68,24%	66,45%
Cosine similarity	63,57%	-
Jaccard similarity	56,96%	55,63%
Jaro-winkler	54,87%	53,79%

Для датасета GSD, кластеризованного с помощью другого метода, а именно DBSCAN, получены такие метрики:

Таблица 10. Полученный процент точности датасета GSD на модели RuPrompts (класт. DBSCAN)

Название метода тестирования	**Средний процент точности**	**Средний процент точности при разбиении на признаки**
Levenshtein distance	29,36%	33,43%
Sequence Matcher	38,59%	41,85%
Cosine similarity	16,46%	-
Jaccard similarity	36,46%	39,13%
Jaro-winkler	57,55%	53,86%

Для датасета Taiga, кластеризованного с помощью DBSCAN, получены следующие метрики:

Таблица 11. Полученный процент точности датасета Taiga на модели RuPrompts (класт. DBSCAN)

Название метода тестирования	**Средний процент точности**	**Средний процент точности при разбиении на признаки**
Levenshtein distance	26,78%	31,1%
Sequence Matcher	36,17%	38,53%
Cosine similarity	17,06%	-
Jaccard similarity	32,73%	34,75%
Jaro-winkler	56,21%	52,63%

Для сравнения мы также обучили одну из моделей нейронных сетей на некластеризованных данных датасета SynTagRus:

Таблица 12. Полученный процент точности датасета SynTagRus на модели RuPrompts

Название метода тестирования	**Средний процент точности**	**Средний процент точности при разбиении на признаки**
Levenshtein distance	62,9%	66,17%
Sequence Matcher	68,16%	74,01%
Cosine similarity	53,15%	-
Jaccard similarity	64,13%	66,95%
Jaro-winkler	80,91%	76,41%

Для нашей задачи мы выбрали метод Jaccard similarity как основной определяющий критерий по нескольким причинам. Данный метод учитывает множества элементов, а не символов. Jaccard similarity вычисляет коэффициент сходства на основе пересечения и объединения множеств. Т. е. в нашей задаче это позволяет учесть, какие признаки слова совпадают (например, "Inan" в паре “NOUN,Inan,Sing,Fem” и “NOUN,Inan,Plur,Masc”) и какие различаются (например, "Sing" и "Plur" в той же паре). Он учитывает и количество общих атрибутов, и общее количество атрибутов в обоих строках. Также метод Jaccard similarity не учитывает порядок элементов в строках, что может сыграть свою роль в нашей задаче.

Сведем все полученные данные в одну таблицу для сравнения результатов тестирования:

Таблица 13. Полученный процент точности с помощью метода Jaccard similarity

Используемый датасет, модель и метод кластеризации	**Средний процент точности**	**Средний процент точности при разбиении на признаки**
GSD, T5, K-means	23,02%	27,54%
Taiga, T5, K-means	68,11%	72,72%
Taiga, T5, AC	69,24%	73,5%
GSD, RuPrompts, K-means	56,96%	55,63%
GSD, RuPrompts, DBSCAN	36,46%	39,13%
Taiga, RuPrompts, DBSCAN	32,73%	34,75%
SynTagRus, RuPrompts, No clusters	64,13%	66,95%

Наибольшую точность предсказаний показывает модель T5, обученная на кластеризованных с помощью метода Agglomerative Clustering данных датасета Taiga.

В целом результаты обучения используемых нейросетей могут варьироваться в зависимости от множества факторов, включая выбранный алгоритм обучения, архитектуру нейронной сети, объем и качество обучающих данных, параметры обучения и другие.

5. Результаты исследования

5.1 Анализ преимуществ и ограничений разработанной модели

В результате нашего эксперимента мы получили следующие результаты:

1. Повышение точности предсказаний:

   Кластеризация данных с помощью метода Agglomerative Clustering позволяет сгруппировать похожие данные в одни кластеры. Это полезно для обучения нейронной сети, поскольку она имеет доступ к более структурированным и категоризированным данным. В результате, модель T5, обученная на таких данных, показала более высокую точность предсказаний.

2. Улучшение обобщающей способности:

   Кластеризация данных помогла выделить общие характеристики и паттерны в данных. Это позволило модели T5 лучше обобщать и переносить свои знания на новые примеры, которые не были представлены в обучающем наборе данных.

3. Сокращение размера данных:

   Кластеризация позволила сократить размер данных, представляя их компактно в виде кластеров. Это привело к снижению потребности в объеме памяти и вычислительных ресурсах для обучения модели T5, что ускорило процесс обучения и сделало его эффективнее.

4. Возможность извлечения более релевантных признаков:

   Возможно кластеризация данных помогла выделить более релевантные признаки и атрибуты для задачи предсказания. Это могло привести к улучшению способности модели T5 распознавать и использовать значимые особенности данных, что в свою очередь могло повысить точность предсказаний.

Однако наша модель имела и свои недостатки:

1. Зависимость от качества кластеризации:

   Результаты модели могут быть сильно зависимы от качества проведенной кластеризации. Неправильное или низкокачественная кластеризация может привести к искажению данных и, в результате, к снижению точности предсказаний модели T5.

2. Возможные проблемы с масштабированием:

   Учитывая то, что исходный датасет Taiga очень большой, то кластеризация может столкнуться с проблемами масштабирования. Обработка больших объемов данных может быть вычислительно сложной задачей и требовать значительных вычислительных ресурсов.

3. Потеря информации при кластеризации:

   Кластеризация данных может привести к потере некоторой информации, особенно если различные кластеры имеют схожие характеристики. Это могло снизить способность модели T5 улавливать тонкие отличия между данными и повлиять на точность предсказаний.

4. Ограниченность обучающих данных:

   Обучение данной модели на кластеризованных данных ограничивается доступными кластерами и структурой данных в датасете Taiga. Возможно, в данных есть скрытые или неучтенные паттерны, которые могут быть полезными для предсказаний, но не учтены в процессе кластеризации. Либо датасет недостаточно разнообразен и наполнен всеми примерами употреблений тех или иных словоформ.

5.2 Лингвистическая ценность кластеризации слов и разработки нейронной сети

Кластеризация слов имеет свое значение в области лингвистики. Во-первых, данный метод группировки данных помогает в идентификации лексических и грамматических групп. Путем группировки таких слов, которые демонстрируют схожие грамматические признаки или лексические характеристики, можно выделить типичные шаблоны и структуры в языковой системе. Это позволяет лингвистам лучше понять организацию и функционирование грамматических и лексических систем языка.

Во-вторых, кластеризация слов может помочь в определении грамматических категорий, таких как род, число, падеж и другие. Группировка слов на основе их признаков позволяет выявить общие морфологические и синтаксические закономерности, которые указывают на существование определенных грамматических категорий в языке. Это сможет способствовать

развитию лингвистических теорий и моделей, а также обогатить лингвистическую типологию.

В-третьих, с помощью кластеризации можно достичь успехов в анализе различных языковых диалектов. Группировка слов по схожим грамматическим и лексическим характеристикам позволяет выявить различия и сходства между разными диалектами языка. Это поможет понять разнообразие языковых систем, а также изучить различные языковые контакты и влияние одних языков на другие.

В-четвертых, создание новых лингвистических ресурсов, таких как грамматических баз данных, электронных словарей и корпусов, возможно с помощью кластеризации. Группировка слов позволяет организовать всю информацию в структурированном и удобном формате, что облегчает поиск лингвистической информации для слов.

Разработка нейронной сети для предсказания грамматических признаков слова, обученная на кластеризованных данных, также приносит значительную пользу с точки зрения лингвистики. Полученная модель позволяет подтвердить или опровергнуть лингвистические гипотезы о грамматической организации языка. Если нейросеть успешно предсказывает грамматические признаки слова, основываясь на кластеризованных данных, это может свидетельствовать в пользу определенных лингвистических теорий или моделей.

Изученные нейросети обладают способностью обнаруживать сложные грамматические паттерны и зависимости, которые могут быть трудно обнаружимы или объяснимы с помощью традиционных лингвистических подходов. Анализ работы данных нейронных сетей может привести к новым открытиям о грамматике и структуре языка.

Результаты работы сети могут помочь идентифицировать общие и отличительные грамматические характеристики различных языков и семей языков. Это может привести к уточнению и расширению существующих лингвистических типологических моделей.

Также использование нейронных сетей может помочь в изучении языковых изменений и эволюции языка. Путем анализа различий в предсказаниях сети для разных периодов или разных географических вариантов языка можно выявить и описать изменения в грамматических системах языка и их причины.

Результаты данной работы могут расширить наши знания о языковых системах, типологии и языковой эволюции, а также привнести новые данные и подтвердить или скорректировать существующие лингвистические гипотезы и модели.

6. Рекомендации для дальнейших исследований

Мы рекомендуем несколько стратегий продолжения нашего исследования:

1. Можно рассмотреть использование других архитектур нейронных сетей либо более больших или других предобученных моделей T5 или RuPrompts. Например, можно попробовать рекуррентные нейронные сети, сверточные нейронные сети либо другие модели трансформеров. Каждая архитектура имеет свои особенности, и эксперименты с разными моделями могут привести к различным результатам.
2. Разные датасеты для обучения и тестирования нейросетей также могут помочь в получении лучших результатов. Можно создать свой датасет морфологически размеченных текстов, содержащий различные жанры или диалекты языков, подходящий под определенную лингвистическую задачу. Это позволит оценить универсальность и обобщающую способность разработанных моделей нейронных сетей.
3. Попробовать другие методы кластеризации, так как каждый метод имеет свои преимущества и ограничения. Эксперименты с разными методами помогут понять, какие подходы лучше всего работают для данной лингвистической задачи.
4. Использовать другие метрики оценки предсказаний нейронной сети для сравнения результатов. Например, можно использовать стандартные метрики, такие как точность (precision), полноту (recall), F-меру, макро- или микро-усреднение (для нескольких классов) и другие метрики для оценки качества предсказаний моделей.

Проведение таких экспериментов позволит получить более обширное представление о различных аспектах разработки морфологических анализаторов на основе данных, размеченных нейросетью. Это позволит оценить

эффективность различных подходов и выбрать наиболее оптимальные решения для конкретных лингвистических задач.

Список литературы

A. B. Сокирко (2010). Быстрословарь: предсказание морфологии русских слов с использованием больших лингвистических ресурсов // Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Международной конференции «Диалог». Вып. 9 (16). М.: Изд-во РГГУ, 2010. С. 450–456.

A. Belonogov, & N. Konstantinova (2021). Neural network-based morphological analyzer for Russian language // Proceedings of the 2021 International Conference on Computational Linguistics and Natural Language Processing (pp. 42-48). ACM.

O. V. Dereza, D. A. Kayutenko, A. S. Fenogenova (2016). Automatic morphological analysis for Russian: a comparative study // Proceedings of the International Conference Dialogue 2016. Computational linguistics and intellectual technologies.

A. Z. Faridee, & F. M. Tyers (2009). Development of a morphological analyser for Bengali. FREEOPMT.

G. Kessikbayeva, I. Cicekli (June 2014). Rule based morphological analyzer of Kazakh language // Proceedings of the 2014 Joint Meeting of SIGMORPHON and SIGFSM, Baltimore, Mary-land, Association for Computational Linguistics

M. Korobov (2015). Morphological Analyzer and Generator for Russian and Ukrainian Languages // Khachay, M., Konstantinova, N., Panchenko, A., Ignatov, D., Labunets, V. (eds.) Analysis of Images, Social Networks and Texts. AIST 2015. Communications in Computer and Information Science, vol 542.

Y. Kuznetsov, & A. Lesin (2018). Neural network-based morphological analyzer // Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (pp. 5084-5094). Association for Computational Linguistics.

A. Panchenko, N. Konstantinova, & N. Loukachevitch (2021). Neural network-based morphological analyzer with the possibility of embedding additional linguistic information // Proceedings of the 2021 International Conference on Information Technology and Systems (pp. 1-8). IEEE.

I. Segalovich (2003). A Fast Morphological Algorithm with Unknown Word Guessing Induced by a Dictionary for a Web Search Engine. MLMTA, pp. 273-280.

A. Sorokin, T. Shavrina, O. Lyashevskaya, B. Bocharov, S. Alexeeva, K. Droganova, ... & D. Granovsky (2017). MorphoRuEval-2017: an evaluation track for the automatic morphological analysis methods for Russian // Proceedings of the International Conference “Dialogue 2017”.

E. Yurevich, & Y. Kuznetsov (2019). Neural network-based morphological analyzer with incremental training // Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (pp. 4274-4279). Association for Computational Linguistics.

I. Dabbura (2018). K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks // Towards Data Science. https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a

E. Lutins (2017). DBSCAN: What is it? When to use it? How to use it // Towards Data Science. https://elutins.medium.com/dbscan-what-is-it-when-to-use-it-how-to-use-it-8bd506293818

https://en.m.wikipedia.org/wiki/Hierarchical_clustering

R. Gandhi (2018). Naive Bayes Classifier // Towards Data Science. https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c

T. Yiu (2019). Understanding Random Forest // Towards Data Science. https://towardsdatascience.com/understanding-random-forest-58381e0602d2

T. Arkhangelskiy (2021). Uniparser morphology. https://uniparser-morph.readthedocs.io/en/latest/

Приложение

Ссылка на github с материалами работы:

https://github.com/Mefeoss/Diploma_MA_with_neural_networks/tree/main