ПРОЕКТ ПО ЛИТЕРАТУРЕ «АНАЛИЗ ТЕКСТОВ»

Владимир Латыпов

ВВЕДЕНИЕ

Иногда на уроках литературы учителя обращают внимание на некоторые общие свойства текстов у одних и тех же авторов. Например, одни авторы любят длинные предложения, другие часто используют парцелляцию. Каждый имеет свой неповторимый "стиль". Этот "стиль" может быть описан большим количеством косвенных признаков, таких как частота использования частей речи, длина и структура предложений e.t.c. В этой работе я собрал большое количество отфильтрованной литературы и попробовал программно выделить и проанализировать некоторые такие признаки.

Из-за отсутствия готовых данных пришлось использовать `web-scraping` (парсинг), используя `Python`. Данные фильтровались от стихов и произведений недостаточного размера. Для оставшейся части из-за требуемого быстродействия использовался язык `C++`. Также для части признаков использовалась технология `word2vec` (имея очень большой объем данных, удалось получить очень качественно натренированную нейросеть).

Такие признаки могут быть использованы для определения автора или стиля по произведению путем кластеризации (это может помочь в решении таких задач, как истинное авторство таких книг, как "Тихий дон" или "Конек-Горбунок"). Возможно, это может помочь с постепенно становящейся все более актуальной проблемой детектирования текстов, сгенерированных нейросетями. Также интересно наблюдать за динамикой произведений на временной шкале. Например, существуют писатели (классический пример

— Александр Сергеевич Пушкин), "опередившие время", то есть своим творчеством задавшие дальнейшее развитие языка. Очень много можно исследовать, используя технологию `word2vec`, позволяющую вникать в суть произведения — вплоть до определения характера героев. Открываются большие просторы для исследований привычной "гуманитарной" области с помощью количественных оценок.

СОДЕРЖАНИЕ

ПОЛУЧЕНИЕ ДАННЫХ

Первый и необходимый этап, которому посвящена первая часть доклада, — получить как можно больше данных для анализа и обучения модели. Для этих целей я использовал сайт "royallib.com", с которого удалось скачать около 50 ГБ чистого текста в формате "txt", отсортированного по авторам.

Для этой части проекта использовался язык `Python`, для остальных задач использовался `C++`. Здесь можно посмотреть код.

Однако это было не просто. Надо иметь структурированные данные, то есть знать, какому автору какое произведение принадлежит — это основа классификации.

Для этого на сайте есть множество разделов, например, список ссылок на страницы авторов, отсортированных по алфавиту. На страницах авторов — ссылки на страницы произведений, на которых есть описание книги, жанр и ссылка для скачивания архива, который содержит иллюстрации и, конечно, сам текст.

Каждый уровень вложенности страниц сайта я обрабатывал последовательно. Но если, посылая запрос серверу, просто ждать, пока он ответит, и ничего не делать, то по моим расчётам на это уйдёт месяц. Но дело здесь не в скорости самого интернет-соединения, а именно в задержке. Это значит, что стоит распараллелить программу, чтобы пока один поток ждал завершения запроса, остальные обрабатывали свои данные. К счастью, сайт не имеет никакой защиты от парсинга. Поэтому я создал 200 независимых потоков. Пришлось обрабатывать множество ошибок, связанных с человеческим фактором, так как какую бы умную систему ты не сделал, найдётся какой-нибудь Вася Пупкин, который напишет такое, что хоть стой, хоть падай... Что бы из этого программа не сделала, меня это не устраивает.

В итоге получилось всё скачать, однако среднее качество этого текста оставляло желать лучшего из-за обилия неизвестных авторов, пишущих низкокачественные произведения. Их можно использовать только для обучения обычных языковых параметров (таких как словарный запас, имена, семантическое значение таких слов, как "зелёный", например, и т.д.), но не чего-то литературного, например, аллитераций и ассонансов в случае стихов или уровня вложенности причастных оборотов или глобального распределения векторов слов во всём произведении в случае прозы.

Итак, для того чтобы отфильтровать авторов, я решил использовать такой критерий, как существование статьи в Википедии об авторе. Для этого использовались эвристики, так как нужно определить не только существование статьи, но и:

1. Принадлежность её именно этому человеку, так как поиск Википедии — штука довольно гибкая и она может, например, найти человека с похожей фамилией, что крайне нежелательно, так как опечаток в данном случае не быть не может. Если фамилия похожая, но не такая, то это жирный минус в копилку того, правда ли, что автор популярный.
2. Принадлежность статьи именно автору литературных произведений, а не политику и не спортсмену. Для этого лемматизируем слова в описании человека и сравним, есть ли среди них те, которые говорят о том, что это автор, например: `["произведение", "автор", "проза", "драматург"]`.

К сожалению, Википедия, как и любой уважающий себя сайт, имеет защиту от парсинга, причём очень непростую. Сначала после большого количества запросов выдавалась ошибка, что сервер отказался отвечать. Способ справиться с этим — сменить IP. Для этого, оказалось, достаточно перезагрузить роутер. Чтобы программа не ломалась и не останавливалась, я запускал отдельный поток, в котором проигрывался записанный мной звуковой файл с сообщением о том, что следует перезагрузить роутер. Но в таком случае пришлось бы это делать каждые полчаса в течении месяца (по моим расчётам).

Я оставил эту часть на некоторое время, решил пытаться парсить, используя большое количество потоков. И тут почти перестали выдаваться те ошибки. Зато стали новые. Теперь она просто не отвечает долгое время. Сколько угодно. Это останавливает весь процесс. К тому же, всё ещё встречаются ошибки разных видов.

В итоге, с помощью продвинутой системы temp-файлов и обработки исключений получилось проверить всех авторов.

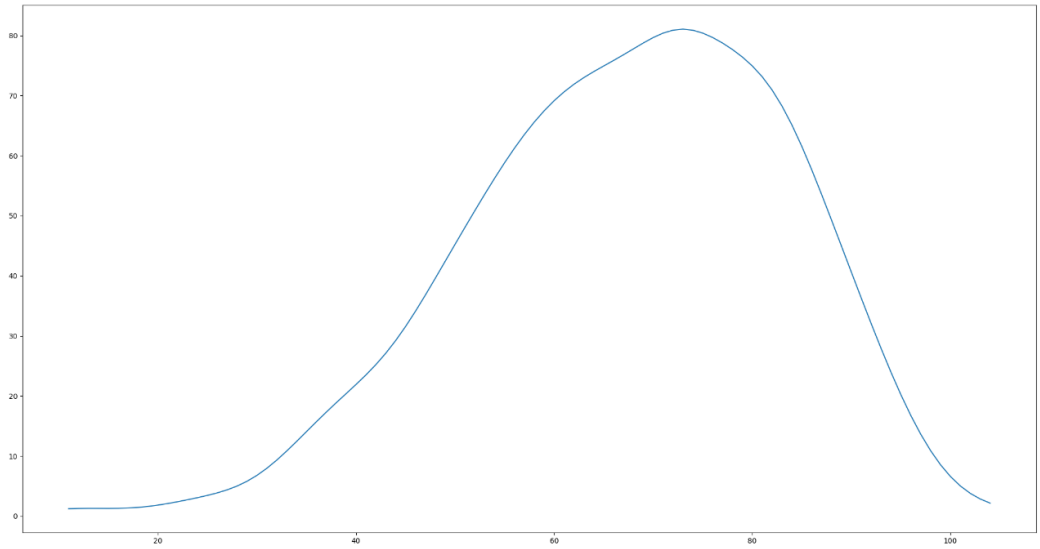
Для каждого из авторов, для которых принято решение о том, что он известный, из Википедии находятся данные о его годах жизни, жанре и т.д. Вот пример полученной информации:


```
[
    {
        "name": "Пушкин Александр",
        "artworks": [...],
        "wiki_data": {
            "life": {
                "alive": false,
                "birth_day": 1799,
                "death_day": 1837,
                "age": 38,
                "precision": true
            },
            "raw_title": "Пушкин, Александр Сергеевич",
            "title": "Пушкин Александр Сергеевич",
            "authority": 2,
            "authoric_words": [
                "поэт",
                "драматург",
                "прозаик",
                "критик"
            ],
            "additional_properties": {
                "Имя при рождении": "Александр Сергеевич Пушкин",
                "Псевдонимы": "Александр НКШП, Иван Петрович Белкин,Феофилакт Косичкин (журнальный), Р., Ст. Арз. (Старый Арзамасец), А. Б.[1]",
                "Дата рождения": "26 мая (6 июня) 1799(1799-06-06)",
                "Место рождения": "Москва, Российская империя",
                "Дата смерти": "29 января (10 февраля) 1837(1837-02-10) (37 лет)",
                "Место смерти": "Санкт-Петербург, Российская империя",
                "Род деятельности": "поэт, прозаик, драматург, литературный критик, переводчик, публицист, историк",
                "Годы творчества": "1814–1837",
                "Направление": "романтизм, реализм",
                "Жанр": "поэма, роман (исторический роман, роман в стихах, разбойничий роман), пьеса, повесть, сказка",
                "Язык произведений": "русский, французский[~ 1]",
                "Дебют": "К другу стихотворцу (1814)"
            },
            "is_famous": true
        }
    }
```


Таких авторов нашлось 2 000. Почти все из них действительно авторы. (Случайная выборка проверена руками).

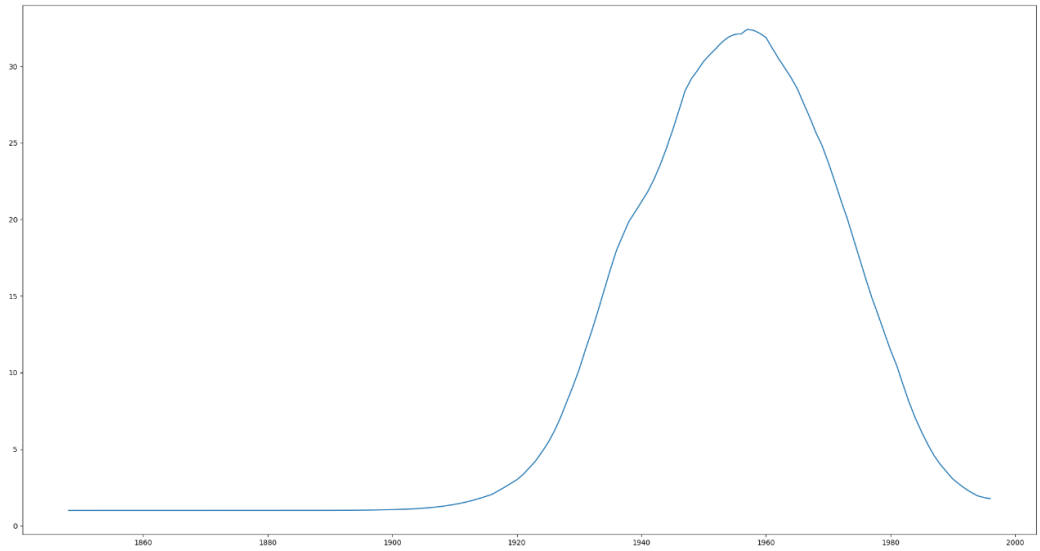
Можно рассмотреть распределение ныне мёртвых авторов по годам жизни (возрасту на момент смерти).

Вот оно. Рассматривайте.



С помощью него можно с хорошей точностью отсеивать поэтов. Они меньше живут.

Также интересно посмотреть на то, когда в основном жили писатели. Опять же, только те, кого уже нет с нами, причём совсем выбросы отфильтрованы
Годы рождения:



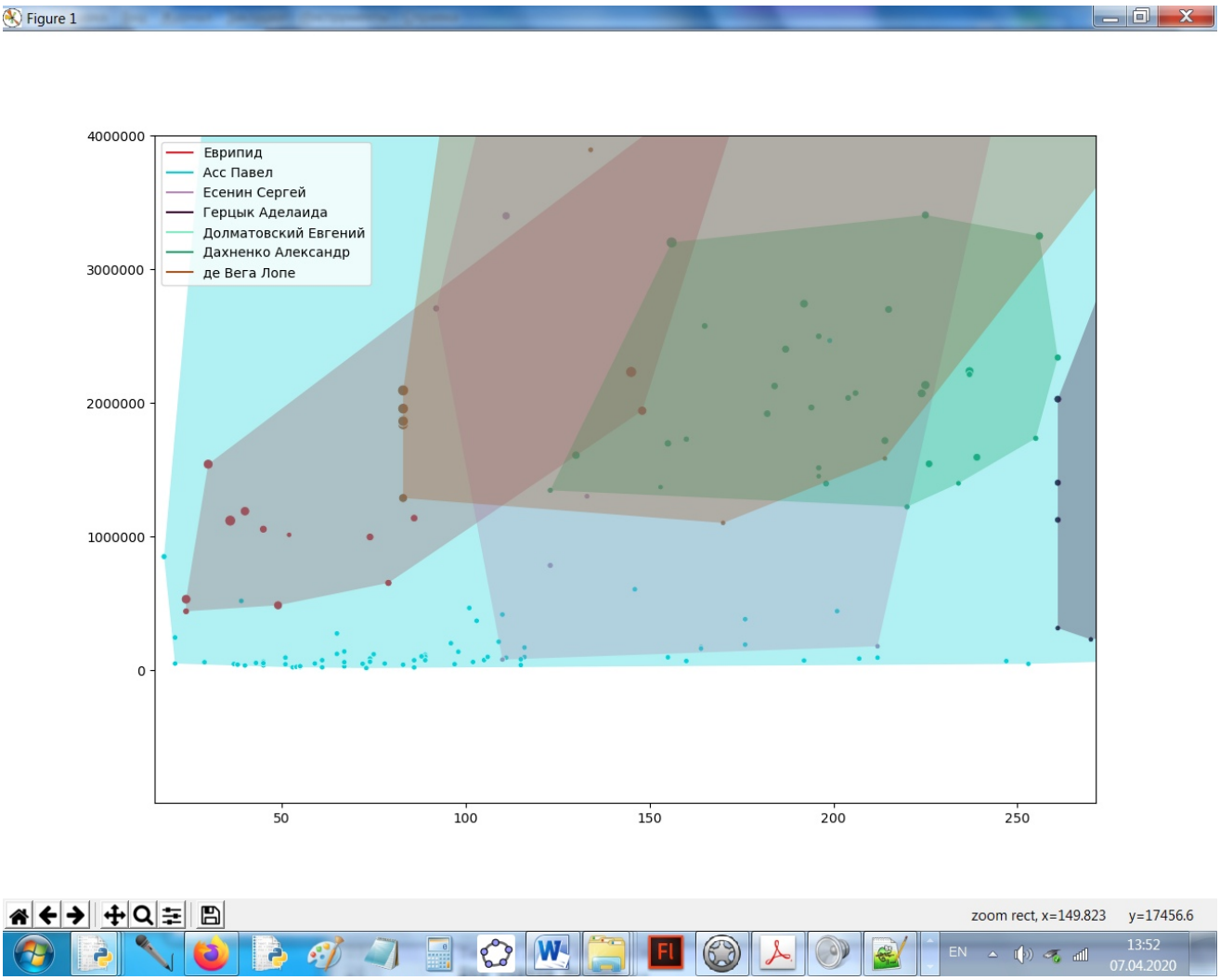
ПОСТАНОВКА ЗАДАЧИ И ФОРМАТ ПРИЗНАКОВ ДЛЯ АНАЛИЗА.

Нужно придумать, какие признаки нужно вытаскивать из произведений, проявляя свои знания предметной области (в данном случае — литературы), реализовать их нахождение, а потом сделать "модель второго уровня", которая, принимая на вход признаки этого(этих) произведен(ия)/(ий), обучается выдавать ожидаемый ответ.

Обычно эти признаки генерируются для каждого произведения по отдельности, таким образом мы получаем пространство высокой размерности, в котором произведения — точки, то есть они описываются координатами, то есть численными значениями каждого из признаков.

Этот подход хорош тем, что в дальнейшем с таким представлением удобно работать, например, классифицировать, производить кластеризацию, поиск ближайших соседей, понижение размерности и другие алгоритмы большим количеством способов.

Вот пример работы такого алгоритма для некоторых произведений:



Видно, что тут происходит перекрытие МВО разных авторов, так как только 2 признака, но 2 — только для визуализации. На самом деле — больше.

Однако, есть и недостатки. К сожалению, далеко не все признаки можно представить в таком виде — то есть в качестве одного или немногих чисел. Например, интересным является распределение предложений по длине. Это не самый, но довольно показательный параметр, причём очень простой в реализации, который после сглаживания представляет из себя набор из 1000 или более точек. В теории можно, например, обучить модель для представления этого графика в виде векторов, но гораздо удобнее и лучше иметь перед собой двух авторов и их распределения. Тогда нам не так важно, в какую сторону они отличаются, так как цель — найти расстояние между авторами. Примерно то же самое и с распределением семантических векторов слов.

Поэтому я выбрал другое целевое представление данных — модель должна оценивать расстояние между данными ей двумя произведениями. Впоследствии можно будет перевести это в векторы посредством вновь изобретённого алгоритма, для которого сначала выбирается целевое количество измерений, генерируются случайные веса всем произведений по каждой координате, а потом посредством многих случайных выборов пар с помощью градиентного спуска и косинусной меры происходит изменение этих весов так, чтобы
расстояния, но уже в пространстве. Пусть и многомерном.

Для обучения, по-хорошему, нужна обучающая выборка, так как хоть генерация признаков программируется людьми, правда, некоторые из признаков — всё равно обучаемые, но главное — модель второго уровня обучать надо обязательно.

Понятно, что в данном случае природа у расстояния экспоненциальная. Это значит, что отношение двух расстояний всегда соответствует тому, как далеко находятся произведения, то есть если рассмотреть распределение логарифма расстояний, то оно будет нормальным. А если анализировать сразу, то возникнет много проблем. Например, разница расстояний на некую константу рядом с нулём и далеко от нуля — это совсем разные вещи. То есть самое интересное происходит близко к нулю. Ещё одна проблема — то, что нейросетевые модели плохо работают с такими данными, так как оперируют не отношениями чисел, а с их линейными комбинациями. Более того, наше пространство — линейное, поэтому могут возникнуть проблемы с правильной кластеризацией.

В итоге — мы будем давать модели случайные пары произведений разных авторов и одного автора, предварительно тщательно отфильтрованные, например, от произведений малого размера, стихов, анекдотов, английских текстов и т.д.

И от модели ожидается некое расстояние, которое зависит от того, один и тот же ли автор это был, а таже, от жанра. Вручную определяется то, насколько какие жанры близки. Жанры известны с сайта royallib.com. И мы учим модель предсказывать то, что от неё ожидается.

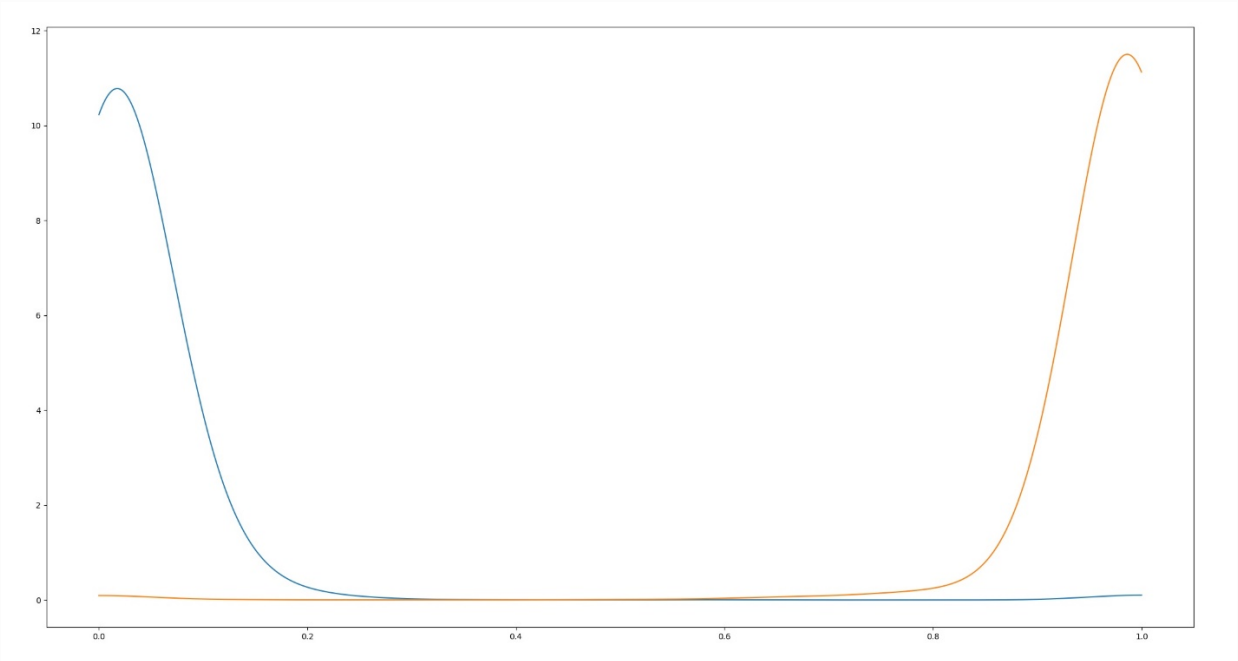
Нужно отметить, что можно позволить иметь много нейронов и почти не бояться *переобучения*, так как количество пар пропорционально квадрату количества произведений, что довольно много.

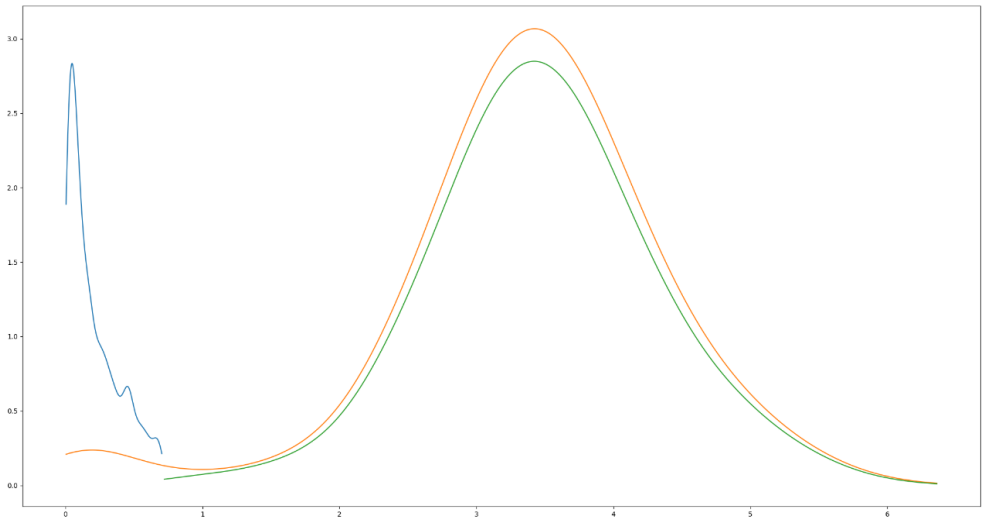
ФИЛЬТРАЦИЯ

Помимо рассмотрения только хороших авторов, нужно ещё много времени посвятить фильтрации произведений, которые мы выбираем для обучения, то есть выбирать не все. Кроме того, для большинства признаков нужно предварительно отфильтровать ненужные слова в тексте и лемматизировать остальные.

1. Критерии того, что произведение стоит рассматривать.

> Для предотвращения помех важно, чтобы оно имело достаточный размер, так как иначе оно будет рассматриваться как полноценное произведение. Для увеличения количества рассматриваемых произведений есть вариант делать "скорость обучения" (learning rate) пропорциональным длине произведения, но всё равно это не позволит рассматривать действительно маленькие произведения. Также, разумеется, не учитываются английские тексты. Вот распределение "английскости" и "русскости" текстов. Видно, что большая часть текстов — русская, но есть и исключения (см. синий холмик справа)



> Стихотворения тоже надо отфильтровывать. Хотя они, скорее всего, и так отфильтруются по длине, их может быть нужно изучать отдельно. Причём именно для предварительной фильтрации не нужно учитывать рифмы и ритм,
> а тем более — ассонансы и аллитерации. В ходе изучения было получено, что наилучшую дифференциацию обеспечивает такой признак, как сумма
> логарифмов [разниц [длин строк длиннее 50 символов] и [константы (50 * 0.9)]]
>
> В итоге так распределяются значения этого признака:
>
> 
>

Оранжевая кривая — всё, зелёная — только проза. Синее — только стихи. Видно, что именно стихи выбиваются из нормального распределения параметра, из чего можно сделать вывод, что этот параметр удачный.

2. Лемматизация слов в каждом тексте

Для начала, очевидно, что для всех параметров кроме грамматических имеет смысл лемматизировать слова, то есть не важно, в какой форме оно стоит. Важно лишь то, какой у него семантическое значение. Чтобы привести слова в начальную форму, конечно, нужно разбирать морфологию слов, но этого недостаточно. Нужно ещё начальную базу для анализа. Для этой цели я использовал файл под названием "Хаген М. Полная Парадигма Русского Языка" размером 200 МБ, в котором хранились разные словоформы и их свойства вроде части речи, рода, склонения и т.д. Удалось также получить список приставок, суффиксов и окончаний с помощью другого файла, а

также
с помощью поиска наибольшей общей подстроки у однокоренных слов.

В итоге получилось создать рекурсивный алгоритм разбора слов по составу, который пытается отрезать от слова известных суффиксов и приставок, но так, чтобы не было такого, что га каком-то фаге отрезаем и получаем из существующего слова несуществующее, например, нельзя отрезать от слова "привет" приставку, так как нет слова "вет". А вот от слова "пришёл" — можно. В итоге алгоритм хорошо работает даже на таком сложном слове, как "изподвыподверта". (Правильно разбирает его по составу)

3. Фильтрация слов в тексте

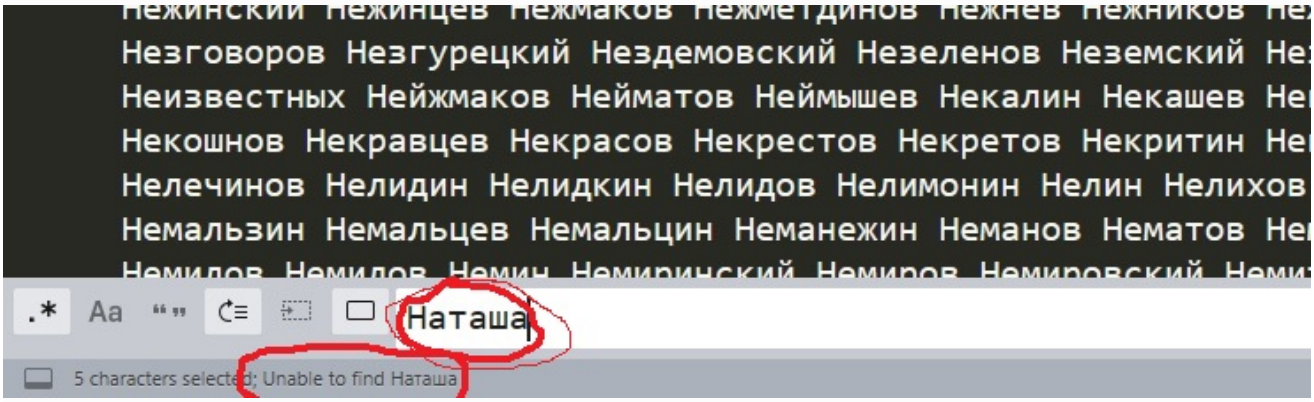
Для минимизации переобучения нужно убирать из текста такие слова как ссылки, имена и фамилии, географические названия, сугубо английские слова и т.д.

Встаёт вопрос, почему это важно и почему это способствует переобучению. Например, у автора есть некий герой, который присутствует сразу в нескольких произведениях или в некоторых частях одного, которые публикуются отдельными книгами. Эта ситуация далеко не теоретическая. Например, Война и Мир у Толстого раздроблена на много частей, а не опубликована целостным произведением. (На целостном находится пометка об
авторском праве... По тому, с какой скоростью правообладатель крутится в гробу, видимо, поняли, что ему это не нравится и решили удалить текст).

Так вот, в этом случае будут 2 произведения, в которых много встречается это имя или географическое название. С другой стороны, для всей базы это редкое слово. Тогда один из признаков будет зашкаливать, так как будет считаться, что у этого слова большой вес. Это и есть переобучение, так как это мешает выявлять истинные корреляции. Более подробно об этой проблеме — в описании признака.

Удаление имён сначала происходило из заранее скачанного словаря. Однако,
он не отличался полнотой. Например, там нет слова "Наташа". А Наталья — есть. Как можно догадаться, я много что тестировал на Войне и Мире,

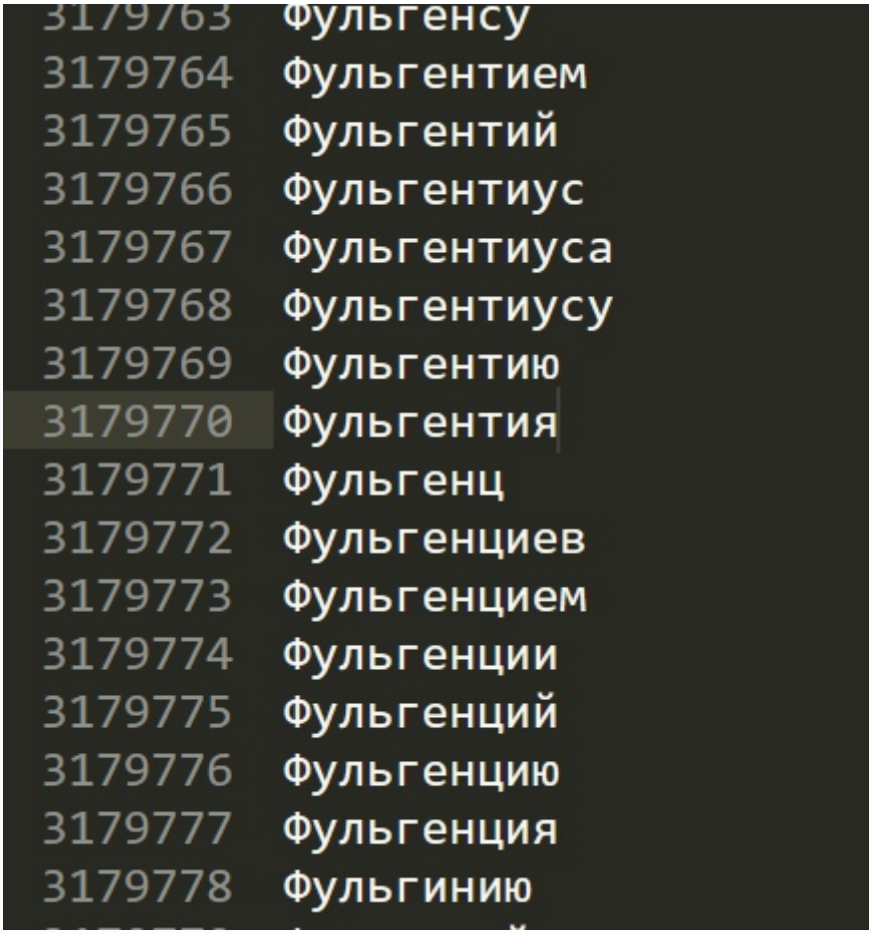
весьма большом произведении.



Чтобы преодолеть эту проблему, я понял, что мне нужно ровно отфильтровывать имена собственные, которые пишутся с большой буквы даже в середине предложения. Также я понял, что если я рассмотрю все слова из всех 50 Гб, то то, что я рассмотрю, будет надмножеством всех анализируемых работ. (Единственное что, нужно пропускать стихи, так как в них иногда пишут начало любой строки с большой буквы). Итак, для каждого слова мы считаем, как чаще оно встречается не в начале предложения, с маленькой, или же, с большой буквы. В зависимости от этого, мы делаем вывод о том, имя нарицательное это, или же Собственное. В итоге поучаем такой большой, исчерпывающий файл. Единственное, что может показаться проблемой, это начало файла (слова отсортированы по алфавиту)

Это вполне ожидаемо (тире — интересный символ с точки зрения разбивания слов. Если он стоит посреди слова, то не делит его, но самостоятельным словом являться не может).

В итоге получился действительно исчерпывающий список со словами, которые я и сам не знал. Вот, например:



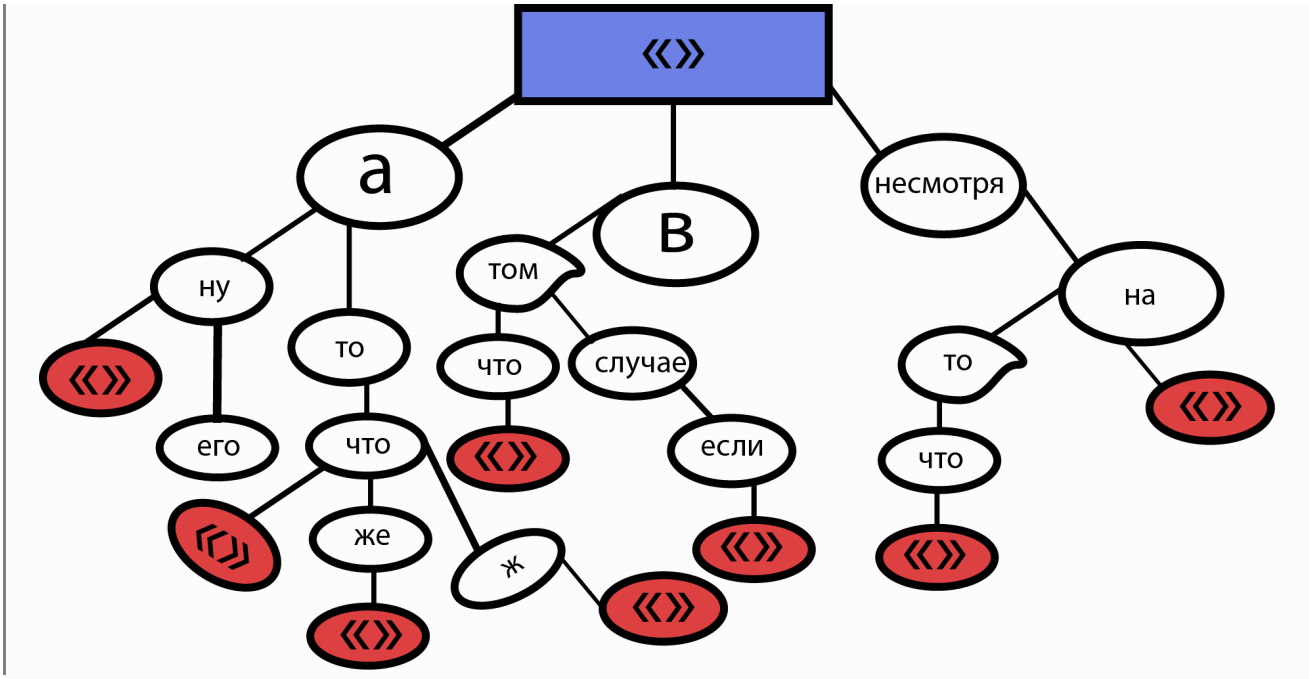
Насчёт Наташи там нет никаких проблем, мягко говоря:

2018433 Наташка-бородавка
2018434 Наташка-вонючка
2018435 Наташка-гармашка
2018436 Наташка-девочка
2018437 Наташка-драчунья
2018438 Наташка-дурочка
2018439 Наташка-ехидна
2018440 Наташка-какашка
2018441 Наташка-коминтерка
2018442 Наташка-мамашка
2018443 Наташка-наташка
2018444 Наташка-неряшка
2018445 Наташка-промокашка
2018446 Наташка-соседка
2018447 Наташка-то
2018448 Наташка-школьница
125609 Андрюлеи
125610 Андрюли
125611 Андрюлик
125612 Андрюлика
125613 Андрюликом
125614 Андрюль
125615 Андрюля
125616 Андрюне
125617 Андрюней
125618 Андрюнису-перунису
125619 Андрюничев
125620 Андрюничева
125621 Андрюничевым
125622 Андрюнька
125623 Андрюнька-горюнька
125624 Андрюню
125625 Андрюня
125626 Андрюс
125627 Андрюс-слатъри
125628 Андрюс-таурус

```
125629  Андрюса
125630  Андрюсе
125631  Андрюсенко
125632  Андрюсик
125633  Андрюской
125634  Андрюското
125635  Андрюсовна
125636  Андрюсовну
```

4. Правильная группировка слов и фильтрация словосочетаний при парсинге.

> Посмотрев на слова, которые определились как неизвестные при парсинге
> и заметив там слово "несмотря", я насторожился. Оказалось, что в собранном словаре у меня было слово "несмотря на", но не "несмотря". Проблему надо было решать.
>
> Это натолкнуло меня на мысль, что вообще хорошо определять словосочетания как одну единицу. Понятно, что плохая идея просто по очереди фильтровать каждое из устойчивых словосочетаний. Это было бы очень долго.
>
> Для этих целей я придумал алгоритм, который впоследствии оказался сильно изменённой версией префиксного дерева, то есть построенного из
> слов, а не букв, а также использованного не по назначению. Выглядит оно так:



Кавычки (кроме верхних) значат, что это словосочетание из пути от корня к этой вершине существует, а узлы без кавычек значат, что к данной строчке добавляется то, что написано на узле. Например, самая правая кавычка соответствует словосочетанию "несмотря на".

При прохождении по всем словам произведения мы в каждый момент находимся
в каком-то месте этого дерева (в начале — находимся в корне). При переходе на новое слово мы смотрим есть ли у этого узла дети с такой строкой. Если есть, переходим на них. Иначе — откатываемся к самому верху и добавляем все слова из пути как отдельные. В теории можно ещё и откатиться на длину пути — 1 и добавить только первое слово, но тогда сложность будет O(n * log(m)), а так — O(n), тем более, мала вероятность того, что в одной фразе будет содержаться другая. Если мы дошли до кавычек, то добавляем всё словосочетание из пути как единое целое.

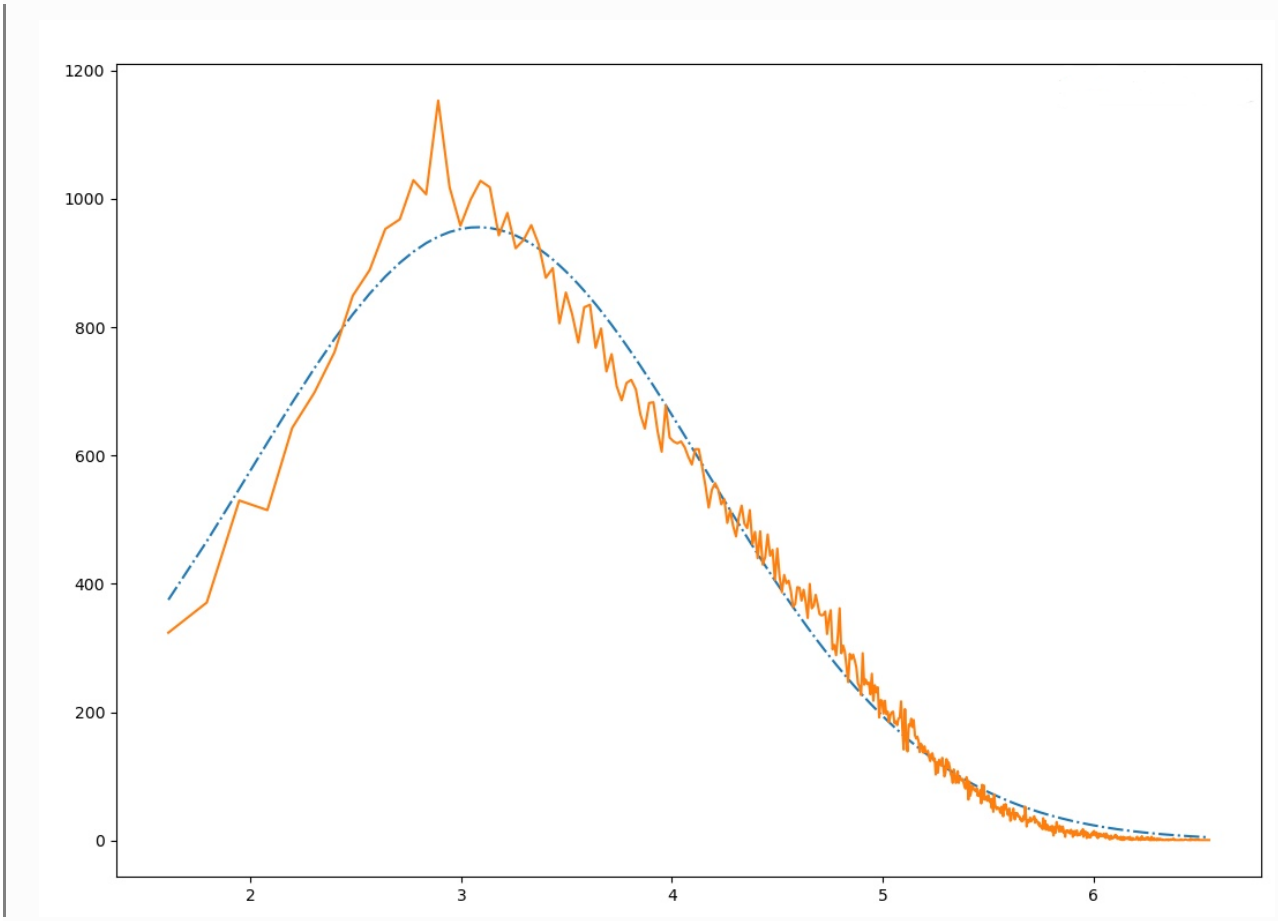
5. Фильтрация "дублей". Это можно делать с помощью разбиения на куски и сравнения количества одинаковых. К сожалению, пока это не реализовано.

ПРИЗНАКИ

Конечно же, важнейшая часть работы — генерация признаков по произведениям.

Распределение предложений по длине.

После логарифмирования оси X выглядит оно так:



То есть распределение логарифмически нормальное. Анализировать его имеет
смыл в таком виде, как показано выше. Для оценки расстояния между произведениями производится сглаживание, а потом происходит подсчёт

суммы квадратов разностей значений полученных функций.

Этот признак имеет смысл. Это можно понять, внимательно почитав произведения Толстого, отличающиеся смещением распределения вправо, в сторону больших предложений. Однако недостаточно брать какое бы то ни было среднее, так как это сводит на нет все индивидуальные особенности автора.

Распределение векторов слов произведения в пространстве.

Тут всё гораздо сложнее. Для начала нужно обучить Word2Vec — векторное представление слов. Оно базируется на том, что похожие слова встречаются в похожих контекстах. Сначала выбирается размерность целевого пространства и происходит случайная инициализация всех параметров слов. Задумка состоит в том, чтобы расстояние между векторами в полученном пространстве отражало то, насколько они непохожи друг на друга. В простом случае можно представить придуманные самостоятельно оси, например,

[

"сила слова ("огромный" сильнее, чем "большой")",

> "размер (например, "корабль" больше, чем "лодка")", "одушевлённость (например, "динозавр" одушевлённее, чем "корабль")"

]

Но. Во-первых, такое количество осей недостаточно для того, чтобы мы, например, понимали, что "корабль" ближе к "лодке", чем самолёт. На практике размерность бывает от 100 до 500.

Во-вторых, измерения не имеют чёткого, понятного человеку смысла, смысл — это только их линейная комбинация, то есть тоже вектор, как и все слова. Характерным примером является тот факт, что если мы вычтем из слова мужчина слово женщина, то получим как бы семантическое понятие "Мужскость". Этот вектор — это теперь конвертер любого понятия из женского в мужской род. Мы, например, его можем применить к слову

"королева", то есть сложить с ним. И получим, что ближайший вектор к сумме обозначает слово "король".

И, разумеется, векторы генерируются и обучаются автоматически, учитывая то, что похожие слова встречаются в похожих контекстах. На каждом шаге мы пытаемся сгенерировать слово на основе контекста, то есть слов, которые стоят вокруг него. Затем, зная правильный ответ, мы обновляем веса.

Есть небольшая проблема. Обучить хотелось бы на всех текстах, однако у меня нет 128 GB оперативной памяти. Пришлось сначала составлять словарь, по нему составлять код Хаффмана, а затем — по очереди обучать по отрезкам размером около 3 Гигабайт.

В итоге — мы имеем для каждого произведения набор векторов, его описывающих. Нужно придумать метрику для сравнения этих наборов. Ультимативным вариантом является построение сглаженных плотностей распределения во множестве контрольных точек, распределённых по пространству. Но проблема в том, что в данном случае есть 100 измерений, то есть требуется порядка 10^{100} вычислений, чтобы по каждому измерению было хотя бы 10 дискрет. Такое вычисление не уложится во время жизни вселенной. Мой вариант — рассмотреть отдельно разные наборы из 5 осей и уже в них не составит труда просчитать и сравнить распределение. Такого количества осей достаточно для того, чтобы выявлять довольно сложные закономерности, а также оно довольно устойчиво к помехам, так как не выявляет лишние закономерности, вероятность появления которых растёт экспоненциально с увеличением числа измерений, то есть закономерностями они и не являются.

Тогда в результате для нахождения расстояния между произведениями мы находим суммы квадратов разностей плотностей распределения векторов в соответствующих точках.

Тут сложно вставить подходящую пятимерную картинку, но должно быть понятно.

Популярные леммы и словоформы.

Векторные представления слов — это богатая идея, но недостаточно. Имеет смысл ещё рассматривать сами корни. Это показала практика, причём не только моя.

Предполагаемый процесс применения признака: нам даны два текста в посчитанными топами слов (каждому слову соответствует его частота появления в тексте). Мы должны выдать расстояние меду двумя текстами. В самом примитивном случае оно равно сумме квадратов разностей частот слов
в текстах для каждого слова.

Но очевидно, что такой подход имеет крайне низкую результативность, так как мы (заметим, что если отбросить слова, не имеющие для нас в данном случае смысла (такие как предлоги, частицы, междометия и т.д.), то в топе будут слова "сказать" и "мочь", "время" и т.д.) будем считать, что одинаковой разность в 100 штук слов как для этих, так и для очень редких, например, для слова "дружина", в то время как на самом деле, второе должно дополнительно иметь больший вес, так как оно редкое во всём корпусе.

Для каждого корня мы легко можем определить его частоту во всей базе.

Тогда мы сможем на базовом уровне понимать, насколько он важен для нахождения расстояния между произведениями. Таким образом, разница между
частотами слова "сказать" становится не очень важна. Вопрос только — в какой степени не очень она должна быть важна? Лучше не гадать, а подобрать эту СТЕПЕНЬ (теперь в прямом смысле). Как и всё обучение в данном случае, оно базируется на генерации пар произведений и предполагаемой дистанции между ними.

Целевая функция — отношение меду неким средним выдачи расстояния для одного и для разных авторов. При подсчёте среднего производится логарифмирование этих расстояний, потом выброс выбросов и подсчёт среднего квадратичного, только вместо степени 2 берётся степень 1.5 и происходит сохранение знака, то есть функция безопасной степени:

*safe_pow (x, exp) = sgn(x) * pow(abs(x), exp)*

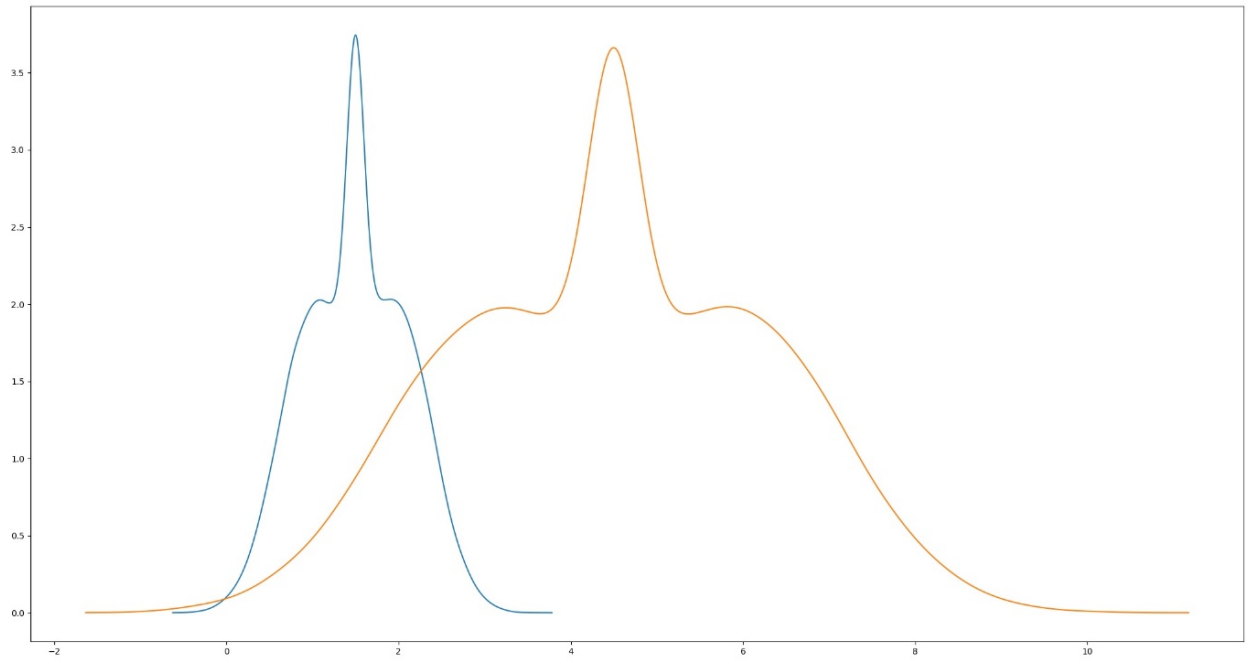
Оптимизация функции происходит с помощью Генетического Алгоритма, про который тоже можно долго рассказывать.

Моя реализация его на C++ находится по этой ссылке.

Характерной особенностью является то, что предполагается, что всё равно вычисление целевой функции занимает бОльльшую часть времени, так что

1) Её надо вычислять многопоточно (что и происходит)

2) Можно применять любые эвристики, даже если они будут не моментальными. Я решил сделать для размножения примерно такое распределение значения генома при определённых значениях у родителей:



Как можно заметить, каждое из них состоит из трёх нормальных. Это делается для того, чтобы каждый потомок мог унаследовать Признаки от каждого родителя. Обычно это достигается другими методами.

Так как происходит большое количество генераций нормально распределённых
случайных чисел, я сделал такую систему для ускорения. Сначала сгенерировать некоторое их количество "по-честному", то есть, например,

через интеграл, для σ = 1, μ = 0, а потом при запросе с сигмой и мю мы берём случайное число из набора, домножаем его на σ и прибавляем μ. Рабочесть такого метода следует из линейности относительно σ и сдвиговости относительно μ.

Это позволило ускорить генерацию в десятки раз.

Также планируется подобрать индивидуальный вес для каждого из популярных слов. К сожалению, пока это не реализовано. Сделать это не очень сложно. Хочется также отметить, что это не будет способствовать переобучению, частой проблеме в этой области.

Склад предложений

Что это значит и как помогает нахождению расстояния между авторами, можно понять на простом примере.

Рассмотрим древнерусски сложенную фразу:

"И пошёл он на войну — Русь-Матушку защищать."

И современную:

"Он пошёл на войну, чтобы защитить Россию"

Среди прочих отличий можно выделить то, что раньше писали сказуемое перед подлежащим, а теперь — наоборот. Именно это и является хорошим критерием.

Разбор предложения по составу

Это тоже может многое сказать о произведении и его авторе. Например, уровень вложенности причастных и деепричастных оборотов. Так как русский язык не однозначен, такой анализ — сложная задача. Для её решения используется представление каждой запятой в одном из 5 видов. Они могут быть открывающимися или закрывающимися (как скобки), при этом одновременно отделять основы предложения либо внутренние обособления. Каждая запятая при таком анализе может одновременно относиться к двум или более различным категориям. Перебираются все возможные варианты.

Вот пример:

Будем обозначать запятую, связывающие 2 однородных члена, выраженных одним словом, знаком "|", запятую, разделяющую 2 основы предложений, символами "[" и "]". А причастные обороты обозначаются фигурными скобками: "{" и "}".

Тогда так будет выглядеть ожидаемая расстановка "запятых":

{Петя | Маша и Ваня (в последний раз наслаждаясь свежим воздухом) играли в футбол} {пока родители смотрели на это}

Такой результат можно получить, перебирая все согласованные с точки зрения баланса скобок варианты их расстановки (таких очень немного), а потом перебираем и смотрим, где наименьший уровень несогласованности. Дальнейшая задача становится элементарной.

Первое, что мы делаем — склеиваем однородные члены в один.

На каждый участок внутри мы смотрим, ищем существительное / местоимение / склейку из предыдущего шага, ищем подходящий глагол. Если мы что-то не
нашли, то эта основа не имеет того, что мы не нашли. Есть ещё редкие частные случаи, например, когда сказуемым является не глагол, но это тема для отдельного исследования.

Частота использования разных частей речи.

Этот признак довольно прост, но оттого не менее эффективен. Особенно хочется выделить глаголы и прилагательные. Частое использование первых говорит о повествовательном стиле изложения, а вторых — об описательном. Технически имеет смысл подавать частоту всех основных частей речи на модель второго уровня и обучить её.

МОДЕЛИ ВТОРОГО УРОВНЯ.

Это то, что принимает на вход описанные выше параметры и обучается находить по ним итоговое расстояние.

Для этого хорошей идеей является использовать перцептрон. Он получает на входе логарифм расстояния, обучается упомянутым выше образом.

Альтернативным вариантом было просто складывать расстояния, возможно, с разными степенями, подобранными с помощью ГА, но это не позволяет выявлять сложные закономерности, хотя они могут быть. Суммирование со степенями всегда возрастает / всегда убывает с изменение определённого параметра в одну сторону.

Например, существуют такие вещи, которые могут позволить себе делать только гении или писатели совсем низкого уровня, поэтому если мы уже уверены, что уровень автора на высоте, то высокое значение этого параметра будет значить, что он гений, а если понятно, что он не очень, то это только ухудшит его оценку.

Конечно, в реальности нет ничего абсолютно категоричного. Всё существует в какой-то степени. Но паттерн, показывающий преимущества перцептрона, выглядит примерно так. Можно ещё заметить, что в свёрточной нейросети происходит то же самое. Свёрткой выявляются признаки, а потом перцептрон их использует, интерпретирует.

ВОЗМОЖНЫЕ СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ.

После обучения модели находить расстояние, можно её применять.

Например, кластеризовать авторов. Для этого я придумал ещё один интересный алгоритм. Нам даны расстояния между авторами. С помощью бинарного поиска мы находим такой порог (после которого считается, что расстояние такое большое, что вершины не связаны, а до — связаны), при

котором наблюдается максимальная скорость изменения количества компонент
связности при фиксированном изменении этого порога. То есть мы максимизируем производную функции количества компонент связности. К счастью, она должна иметь довольно простую форму, поэтому для её максимизации достаточно использовать пятинарный поиск. При желании после
изначального приближения можно определить функцию ошибки как, опять же, отношение [среднего расстояния внутри компонент связности и расстояния вне их], умноженное на количества компонент в квадрате. А потом перебором ближайших вариантов найти её оптимум.

Эту систему можно использовать, например, для рекомендации книг, выяснения подлинности авторства книг, а также — для поиска похожих литературных стилей среди известных и не очень писателей или находить писателя, который имеет литературный стиль, очень похожий на стиль пользователя.

В качестве примеров работы программы, понятного человеку, я приведу вот эти:

Слова, которые особенно характерны для автора и являются его отличительной чертой.

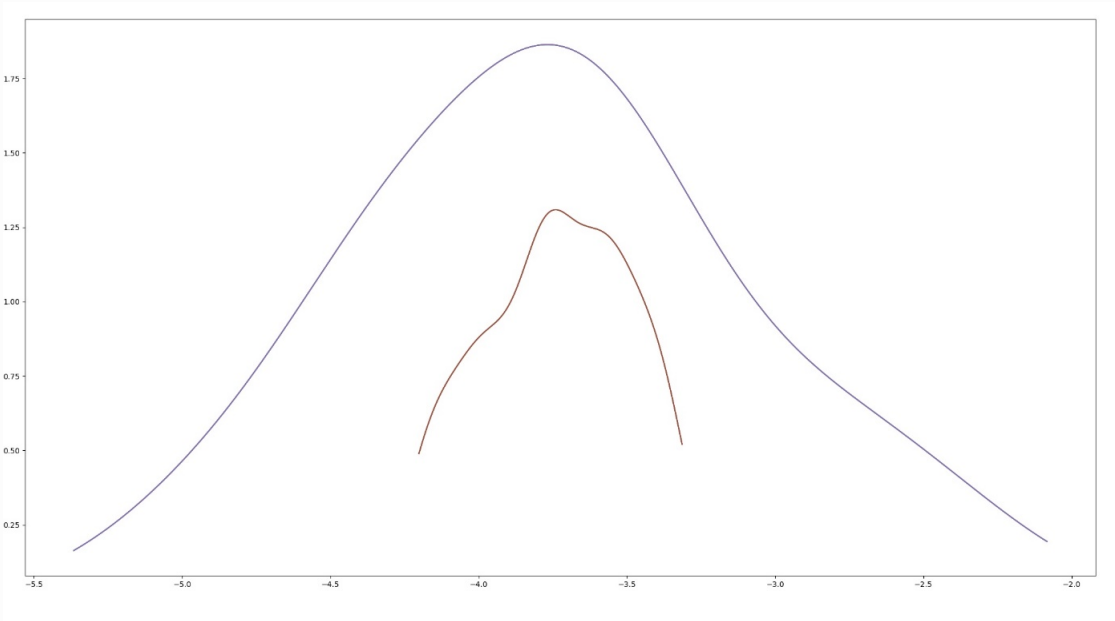
Для этого необходимо не только составить словарь для самого автора, но и седлать это глобально, для всей базы, так как ***характерное*** слово
должно часто встречаться у автора, в то время как в глобальных масштабах быть как можно боле редким. После того, как я проанализировал в сумме более 3 миллиардов слов, глобальный словарь был составлен. Ещё при анализе важно, чтобы слово не просто фигурировало в одном произведении автора.

Теперь встают два вопроса.

1) Как правильно находить частоту слова у автора, чтобы учесть то, что оно должно быть во многих произведениях?

В идеальном случае это будет среднее геометрическое. Но проблема в том, что могут быть выбросы. Более того, распределение концентраций слов — логарифмическое, а среднее арифметическое логарифмов — это всё равно что среднее геометрическое. Это наталкивает на мысль об эффективности работы именно с логарифмами. После взятия логарифма у всех чисел мы находим у распределения сигму, отсеиваем те, которые отличаются от медианы более, чем на n * sigma, где n, подобранное экспериментально, = 0.7. После этого находится взвешенное среднее логарифмов концентраций с весами, равными квадратам длин произведений. И в ответ выдаётся exp(log_mean).

На этой картинке приведена визуализация отсеивания логарифмов по 0.7 * sigma. (Рассматриваются слова "я" в произведениях Толстого).



2) Следующая часть — нахождение индекса характерности слова для автора по найденной средней концентрации и глобальной концентрации в словаре. В идеальном случае — это просто отношение этих концентраций.

Важно, чтобы не попадал всякий мусор, при котором счётчик зашкаливает из-за того, что они мало встречаются в целом, поэтому сразу отсеиваются слова с глобальной концентрацией меньше десяти тысяч. После этого обе концентрации возводятся в некую степень, каждая в свою, и находится отношение. Тут важно чувствовать баланс между всяким мусором, шумами и слишком частыми словами. Думаю, понятно, при слишком большой какой степени что возникает.

И, конечно, вот результаты. Мне интересно было проанализировать Толстого и Достоевского. Результат был показательным:

1) Достоевский:

- помешанный
- исступление
- уродить
- деспот
- обманщик
- присяжный
- подготовительный
- благородие
- виновность
- вскричать
- пролетарий
- беспомощность
- городовой
- убивец
- каторжный
- острог
- мастеровой
- смирный
- предчувствовать
- низость
- игорный
- учащийся
- совестно
- альманах
- артель
- каторга
- неуважение
- вексель
- подлец
- донельзя
- цензор
- барыня
- подсудимый
- фальшь
- развратный
- превосходительство
- словцо
- мещанин
- теперешний
- подло
- брудершафт

- буржуа
- занавесить
- ревизор
- бакенбарда
- декабрист
- тяжба
- писарь
- атеист
- припадок
- раздражительный
- страдалец
- тягость
- ведомость
- куш
- застрелиться
- запрещение
- браниться
- дорасти
- интеллигентный
- струсить
- чрезвычайный
- изверг
- бесчестный

Видно, что удалось отфильтровать весь шум, которого было очень много.

Также видно, что тут есть как характерные слова для старого времени (такие как `писарь, барыня, городовой, превосходительство`, *ещё много*)

Но большую часть характерных слов составляют именно те, которые характеризуют слова, которые очень хорошо описывают мрачный характер произведений Достоевского: {помешанный, исступление, застрелиться, острог, каторжный, развратный, припадок} и так далее.

Есть и другие слова, которые я бы охарактеризовал как присущие интеллигентному писателю, например, {интеллигентный, подготовительный}. Я не имею в виду, что это какие-то умные слова. Дело в том, что большая часть писателей имеют ... более простой стиль.

Также хотелось бы отметить слово пролетарий. Оказывается, Достоевский в ряде книг писал о коммунизме, что его получится достичь только через кровь и страдания:

> Это общество, на основаниях научных, чистая фантазия, что они представили себе человека совсем иным, чем устроила его природа; что человеку трудно и невозможно отказаться от безусловного права собственности, от семейства и от свободы; что от будущего своего человека они слишком много требуют пожертвований, как от личности; **что
> устроить так человека можно только страшным насилием и поставив над ним
> страшное шпионство и беспрерывный контроль самой деспотической власти.

2) Толстой:

- Редут
- князь
- гусар
- опьянить
- тождественный
- увидать
- счастие
- фарисей
- графиня
- вышеупомянутый
- денщик
- главнокомандующий
- сиятельство
- смотр
- который
- порабощение
- нынче
- гусарский
- матка
- исповедание
- истинность
- божеский
- похоть
- вотчина
- соблазн
- ликер
- полковой
- неприятель
- княгиня
- заигрывать
- взглядывать
- приказчик

- рюмочка
- масон
- дурно
- ротный
- барабанщик
- заповедь
- батарея
- адъютант
- трактат
- цензурный
- драгун

Заметим, что средняя длина частых слов у Толстого меньше. Это соответствует действительности и тоже отражает разницу стилей.

Понятно, что тут много слов, характерных для того времени, заметим, что больше, чему у Достоевского, так как Льва Николаевича не такой ярко выраженный зловещий стиль. Но, возможно, имеет смысл делать поправку на время чтобы лучше выяснять именно персональные особенности авторов. Для этого можно составить и сгладить (интерполировать) топ слов для данного времени и сравнивать с ним. Это может быть очень полезной информацией. Интересно смотреть на динамику характерных слов в этом топе.

Тут, конечно, много слов, связанных с ведением сражений в те времена.

Сложно не заметить и то, что здесь много слов, о спиртном и похоти. Люди, глубоко разбирающиеся в литературе, говорят, что Толстой — писатель плоти. Возможно, этот факт отражается на популярных словах.

Хочется отметить иногда появляющиеся слова вроде `тождественный`, `вышеупомянутый`, которые напоминают нам, что Толстой иногда любит пофилософствовать.

Бросается в глаза слово `который`. Оно тоже может быть связано с философствованиями. А возможно, он просто им злоупотребляет.

3) Стругацкие:

- лучевой
- дон
- невесомость
- перегрузка
- генетик

- десантник
- локатор
- бархан
- перепонка
- теорема
- сопка
- остроумно
- программист
- социологический
- кожух
- диспетчер
- утопленник
- дубль
- танкер
- пятак
- 60-х
- интерком
- генетик
- муравейник
- квалифицированный
- осьминог
- сталкер
- саркофаг
- инфракрасный
- скальпель
- кислородный
- уродец
- стервятник
- лучевой
- вепрь
- спрут
- докладчик
- детонатор
- кают-компания
- штурман
- бармен
- галоша
- очкарик
- шлагбаум
- субмарина
- стажер
- бобр
- щебень
- бакенбарда
- марсианский
- сказать
- геолог

- фантаст
- пиявка
- акация
- старикашка
- пластырь
- перископ
- кальмар

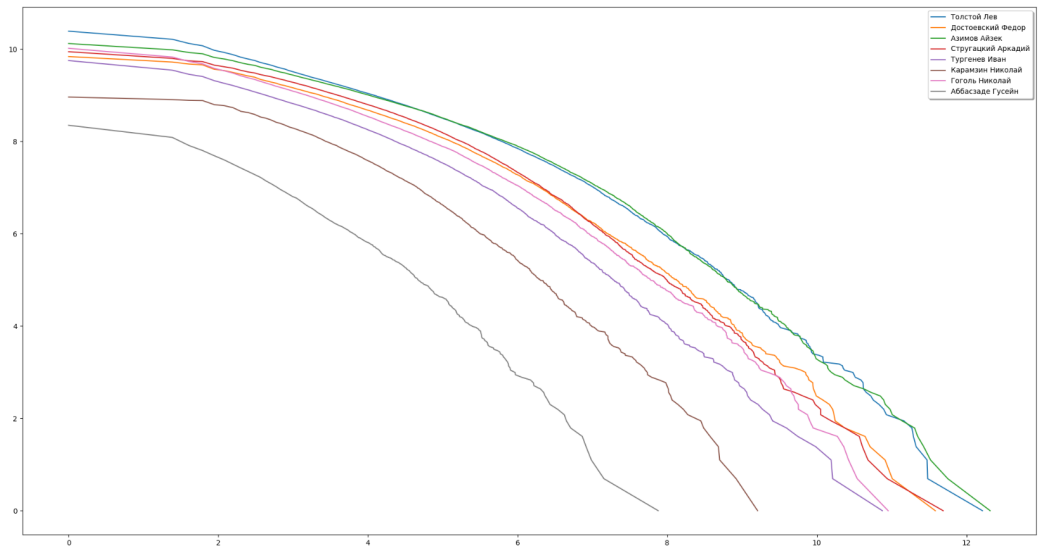
Ну тут как бы без комментариев. Всё и так очевидно.

Единственное что чувствуется, что это всё-такие советские авторы, а не из будущего и что есть слово "сказать" в топе.

Словарный запас разных авторов.

Для этого можно использовать такую метрику: строим график зависимости логарифма количества разных слов с частотой встречаемости выше заданной от логарифма этой частоты.

Проанализировав разных авторов, получим такой график:



Насчёт автора по имени `Аббасзаде Гусейн` всё понятно, хотя и написал он довольно много произведений. Особенно хотелось бы отметить Карамзина и его почти горизонтальную линию вначале. Это станет понятным, если вспомнить, что он жил в допушкинсике времена. Это косвенно подтверждает гипотезу о том, что Пушкин — новатор нашего языка, который ввёл много новых слов, впоследствии пользовавшихся популярностью у авторов.

ССЫЛКИ

1) Скачиватель литературы
2) Генетический алгоритм
3) Модель языка
4) Сам анализатор