Аннотация

Задача идентификации авторов для рукописных текстов является актуальной задачей в области компьютерного зрения, заключающейся в определении количества авторов набора рукописных документов и их кластеризации по писателям. Данная работа посвящана решению данной задачи путем использования различных архитектур сверточных нейронных сетей, а также различных методов кластеризации. В результате работы было протестировано множество моделей и методов, и предложены улучшения полученных результатов.

Содержание

1 Введение **4**

1.1 Актуальность 4

1.2 Постановка задачи 4

1.3 Обзор существующих методов 5

2 Разработанные решения **6**

2.1 Препроцессинг и агрегирование 6

2.1.1 Детекторы углов 6

2.1.2 Агрегирование локальных фрагментов: VLAD 7

2.2 Обучение энкодера 8

2.2.1 Auto-encoder 8

2.2.2 Сиамская нейронная сеть 8

2.2.3 Обучение на задаче классификации 9

2.3 Обучение на синтетическом датасете 10

2.4 Уменьшение размерности 11

2.5 Кластеризация 12

2.5.1 Алгоритмы кластеризации 13

2.5.2 Определение количество кластеров 15

3 Результаты проведения экспериментов **18**

3.1 Метрики качества 18

3.2 Эксперименты 18

4 Заключение **19**

1 Введение

1.1 Актуальность

Идентификация авторов рукописных текстов является актуальной задачей в области компьютерного зрения. Среди сфер использования данной технологии можно выделить анализ исторических документов, обработку рукописных текстов в судебной практике, кластеризацию огромного количества рукописных текстов в образовательной сфере, а также разметку датасета по писателям в автоматическом режиме, что может помочь улучшить качество работы генеративных нейронных сетей, обученных на рукописных текстах.

1.2 Постановка задачи

Работы по данной теме выделяют онлайн и оффлайн методы рапознавания авторов. Онлайн метод подразумевает обработку рукописного текста, который представлен в виде временных фрагментов штрихов, из которых извлекается уникальная информация о писателе. В свою очередь, оффлайн метод проводит анализ изображения уже написанного рукописного текста, из которого извлекаются признаки, по которым выявляется автор.

Задачу идентификации авторов можно решать в постановке как задачи классификации, так и задачи кластеризации. В случае задачи классификации каждый автор представляется из себя отдельных класс, который модель предсказывает, имея на вход рукописный текст. В случае задачи кластеризации, не зная заранее множество авторов и их количество, рукописные фрагменты разбиваются на кластера, каждый из которых предположительно написал один человек. Стоит отметить, что если задача решена в постановке кластеризации, то она решена в постановке классификации, так как в случае успешной кластеризации, можно сопоставить полученные кластера уже известным классам.

Данная дипломная работа будет решать задачу идентификации авторов в формулировке оффлайн кластеризации. Имея на входе документы с рукописным текстом, нужно определить количество писателей и кластеризовать тексты по авторам. Документы могут из себя представлять как полноценные тексты на бумаге, так и отдельно написанные от руки слова или предложения. Обученной модели на стадии inference могут подаваться тексты писателей, которых она не видела во время обучения.

1.3 Обзор существующих методов

Исследования в области идентификации авторов рукописных текстов проводились в течении многих лет, и улучшали постепенно результаты, предлагая различные методы и идеи извлечения и обработки признаков рукописного текста.

Для выявления признаков из полученного изображения современные научные работы в основном делают выбор на сверточных нейронных сетях. Используются различные архитектуры, включая ResNet-18 [источник], ResNet-50 [источник], VGG [источник]. Данные модели показали хорошие результаты в классификационной постановке задачи, где их применяли в качестве энкодеров.

Научные работы предлагают различные варианты обучения энкодера. Например, некоторые исследования обучают энкодер в паре с полносвязной нейронной сетью, используя функцию потерь CrossEntropy [источник]. Также есть исследования в области применения сиамской архитектуры обучения энкодера на данной задаче [источник].

Существует также несколько способов извлечения фрагментов из рукописного текста для дальнейшего извлечения признаков. Один из самых простых способов заключается в нарезания рукописного текста на слова или просто на фрагменты определенной ширины [источник]. В некоторых работах из рукописного текста извлекаются самые информативные элементы почерка, которые обнаруживаются различными алгоритмами обнаружения углов (corner-detectors), например, HARRIS и FAST. После прохождения через сверточную нейронную сеть, полученные эмбеддинги потом агрегируются различными способами. Например находится среднее арифметическое векторов или используется алгоритм агрегации VLAD.

В целях значительного увеличения датасета и, в последствии, улучшения качества обучения, существует идея синтетической генерации датасета рукописных текстов, используя шрифты, похожие на рукописный текст, и применяя аугментацию [источник].

Также, в исследованиях распознования лиц применяется техника обучения Metric Learning, которая помогает в той области получить более репрезентативные эмбеддинги. Так, используя функцию потерь ArcFace удалось достичь значительного улучшения результата в задачи классификации фотографий лиц людей [источник]. Не исключено, что данный метод хорошо себя может показать и на рукописных текстах.

2 Разработанные решения

Исходя из вышеописанных работ, можно составить общую архитектуру решения поставленной задачи. Рукописные тексты сначала проходят через стадию предобработки, во время которой улучшается качество самого рукописного текста, а также происходит его разбивка на фрагменты, либо путем нарезания на слова/части одинаковой ширины, либо путем применения алгоритма нахождения углов для получения максимально репрезентативных элементов почерка. Далее, эти фрагменты поступают в энкодер, который представляет из себя сверточную нейронную сеть, в результате чего получаются эмбеддинги. Далее, эти эмбеддинги при необходимости агрегируются в глобальный эмбеддинг фрагмента текста, если ранее был применен corner-detector. Наконец, применяется алгоритм уменьшения размерности эмбеддингов для улучшения качества кластеризации и применяется сам алгоритм кластеризации.

2.1 Препроцессинг и агрегирование

На вход энкодеру не подается целое изображение документа рукописного текста, так как в нем может содержаться лишняя информация, и энкодеру может быть сложно извлечь репрезентативные признаки из него.

Одним из самых простых решений этой проблемы является нарезание текста на фрагменты одинаковой ширины. Если учитывать, что высота одной строки текста одинакова, то при обучении сети не придется менять размер фрагментов или применять паддинг, чтобы их объединить в батч, тем самым сохраняя все информацию, содержащуюся во фрагменте, и не допуская смещения модели во время ее обучения.

2.1.1 Детекторы углов

Однако даже в уже нарезанном фрагменте может содержаться лишняя информация, так как прямые линии, содержащиеся в почерке, и пустые элементы на бумаге не содержат много информации, по которой можно различить автора. В связи с этим можно энкодеру подавать только фрагменты, содержащие самую важную информацию, например, углы и персечения. Найти подобные участки могут помочь так называемые детекторы углов. Существует множество алгоритмов в данной области. Самыми классическими являются Harris и FAST [источник].

2.1.2 Агрегирование локальных фрагментов: VLAD

После того, как энкодер обработал куски рукописного текста, на выходе мы имеем множество локальных эмбеддингов. Существует несколько способов получения глобального вектора, содержащего репрезентативные признаки текста.

Одним из алгоритмом является VLAD: Vector of Locally Aggregated Descriptors [источник]. Данный алгоритм позволяет получить из локальных дескрипторов общий глобальный вектор фиксированного размера, который содержит достаточно информации для идентификации изображния. Сначала формируется словарь визуальных слов $C = \{c_1, \ldots, c_k\}$ фиксированного размера k с помощью алгоритма кластеризации K-Means, где $k \in \mathbb{R}$ – гиперпараметр. Затем каждому локальному дескриптору x_i сопоставляется ближайшее визуальное слово. Наконец, глобальный эмбеддинг $v \in \mathbb{R}^{k \times d}$ вычисляется по данной формуле:

$$v_{i,j} = \sum_{x \text{ такие что } NN(x)=c_i} (x_j - c_{i,j}) = \sum_{i=1}^{N} a_k(x_i) \cdot (x_i - c_{i,j})$$

где d – размерность пространства локальных эмбеддингов.

Однако, недостатком такого метода является его недифференцируемость. Его нельзя сделать частью модели, которая будет обучаться на картинках рукопинсых текстах. Таким образом, во время обучения мы не можем обучить именно глобальные эмбеддинги, и мы должно будем полагаться на алгоритм VLAD, чтобы он дал глобальные вектора, обладающие нужными нам геометрическими свойствами кластеризуемости.

Для решения данной проблемы существует обновленная версия данного алгоритма называемая NetVLAD [источник]. Он представляет из себя слой, который сопоставим с любой сверточной нейронной сетью, полученный путем замены недифференцируемой операции соотношения локальных дескрипторов визуальным словам на *мягкое* присваивание сразу нескольким кластерам:

$$a_k(x_i) = \frac{e^{-\alpha||x_i - c_k||^2}}{\sum_{k'} e^{-\alpha||x_i - c'_k||^2}}$$

где α – гиперпараметр, $a_k(x_i) \in (0, 1)$, наибольший вес. При $\alpha \to +\infty$ NetVLAD стремиться вести себя аналогично оригинальному алгоритму VLAD.

Таким образом мы сможем обучать аггрегирование локальных эмбеддингов таким образом, чтобы глобальный вектор обладал нужными для нас свойствами, которые помогут дам кластеризовать по авторам более точно.

2.2 Обучение энкодера

Мы поняли как подать сверточной нейронной сети изображение, чтобы на выходе получить вектор. Но теперь нужно обучить энкодер выдавать именно репрезентативные и кластеризуемые эмбеддинги. Для этого существует несколько способов, которые будут описаны далее.

2.2.1 Auto-encoder

Во время выполнения данной работы мы хотели добиться минимального использования меток авторов во время обучения модели. Использование архитектуры автоэнкодера является одним из самых простых способов получения данного результата. Автоэнкодер представляет из себя комбинацию двух сверточных нейронных сетей, одна из которых называется энкодером, а вторая декодером. Данная архитектура обучается на задаче восстановления изображения, которое подается в начале модели. Предполагается, что после обучения энкодер научится ”сжимать” подаваемое на вход изображение в промежуточное состояние, представленное в виде вектора определенной размерности, таком образом, чтобы имеющийся декодер умел уже восстанавливать исходное изображение. Соответственно, промежуточное состояние содержит достаточно информации для восстановления изображения, и, в силу гладкости нейронной сети как обычной математической функции, можно построить гипотезу, что эмбеддинги одинаковых почерков должны находится близко друг к другу.

2.2.2 Сиамская нейронная сеть

Другая идея обучения энкодера исходит из того факта, что мы хотим получить именно кластеризуемые эмбеддинги. Это значит, что эмбеддинги текстов одного автора должны находиться на максимально близком расстоянии, а эмбеддинги различных авторов – на далеком, чтобы потом алгоритм кластеризации смог отделить ”облака” векторов. Существует множество методов обучения нейронных сетей, которые непосредственно закладывают вышеописанное свойство в процесс обучения. Одним из таким методов является архитектура сиамской нейронной сети (SNN). Она представляет из себя пару идентичных нейронных сетей, веса которых непосредственно связаны. Во время обучения подаются пары изображений как от одного автора, так и от разных авторов. Сеть обучается таким образом, чтобы евклидово расстояние между векторами текстов от одного автора было минимально, а между векторами от разных авторов – как минимум равнялось какому-то гиперпараметру α.

Если говорить формально, определим функцию потерь для данной сети следующим образом. Пусть x_i, x_j – изображения почерка, которые подаются SNN на вход, c_k – множество изображений от автора с номером k, α – числовой гиперпараметр. Функцией отсечения назовем:

$$\text{ReLU}(y, \alpha) = \begin{cases} y, & 0 \leq y < \alpha \\ \alpha, & y \geq \alpha \end{cases}$$

Она будет использоваться в формуле функции потерь. Мы не хотим наказывать нейронную сеть во время обучения за то, что эмбеддинги находятся слишком далеко. Поэтому если расстояние между ними будет больше α, то мы будем отсекать его по заранее заданному гиперпараметру.

Определим целевую функцию:

$$Target = \begin{cases} 0, & x_i, x_j \in c_k \\ \alpha, & x_i \in c_k \text{ and } x_j \in c_q \end{cases}$$

Если изображения принадлежат одному автору (находятся в одном множестве c_k), то расстояние между ними должно быть равно нулю. Иначе, если они от разных авторов, то расстояние должно быть как минимум α.

Наконец, определим функцию потерь:

$$\text{Loss}(Target, emb_1, emb_2) = ||\text{ReLU}(||emb_1 - emb_2||, \alpha) - Target||$$

Как мы видим, перед тем как сравнивать расстояние между эмбеддингами и значение target, мы производим отсечку, чтобы не наказывать сеть за слишком далекие друг от друга эмбеддинги.

2.2.3 Обучение на задаче классификации

Альтернативной идеей обучения энкодера является обучение его на задаче классификации, с последующим отсечением классификатора. Стандартной архитектурой при обучении на задаче классификации является связка сверточной и полносвязной нейронных сетей. Можно простроить гипотезу, что эмбеддинги, которые выдает энкодер во время обучения, обладают геометрическими свойствами, которые позволяют классификатору понять к какому автору рукописный текст действительно относится.

Функция потерь в данной ситуации играет огромную роль, так как именно от нее зависит каким образом полученные эмбеддинги будут расположены в пространстве. Обычно при обучении классификатора применяют стандартную функцию SoftMax:

$$L_{\text{SM}} = -\log\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{N} e^{W_j^T x_i + b_j}},$$

где $x_i \in \mathbb{R}^d$ – эмбеддинг i-го изображения от автора с номером y_i, d – размерность пространства эмбеддингов, W и b задают линейное преобразование. Однако, данная функция потерь никак не способствует близости эмбеддингов от одного автора и дальности эмбеддингов из разных классов, и при большом количестве авторов пространство векторов будет плохо кластеризуемо [источник].

Поэтому в области распознования лиц используют другую функцию потерь для задачи классификации [источник].

$$L_{\text{ArcFace}} = -\log\frac{e^{s\cos(\theta_{y_i}+m)}}{e^{s\cos(\theta_{y_i}+m)} + \sum_{j=1, j\neq y_i}^{N} e^{s\cos(\theta_j)}}$$

При обучении с функцией потерь ArcFace эмбеддинги распространяются по гиперсфере радиуса s, и меняется именно угловое расстояние между ними. При этом обеспечивается интервал с геодезическим расстоянием m между эмбеддингами разных классов. Это свойство может дать хорошо кластеризуемые вектора.

2.3 Обучение на синтетическом датасете

Существует множество датасетов, содержащих рукописные тексты. Самые популярные из них это датасет CVL [источник] и IAM [источник]. В общей сложности они содержат рукописные тексты от порядка 1000 авторов. Тем не менее, такого количества данных может быть недостаточно для получения хороших результатов обучения крупных сверточных нейронных сетей из-за проблемы переобучания.

Есть несколько способов решения этой проблемы. Один из них заключается в аугментации тренеровочных данных. Говоря конкретнее, можно изменять текстуру бумагу, силу нажатия и другие параметры почерка, который подается в нейронную сеть для обучения.

Но данный способ не решает проблему количества авторов, также как и разнообразности почерков. Поэтому существует другой вариант генерации самого датасета [источник]. Можно отобрать порядка 10000 шрифтов, которые похожи на рукописный текст, применить к ним аугментацию, например, симулировать написание ручкой или чернилами, добавить различных текстур бумаги и освещения, и сгенерировать изображения из 10000 случайных

слов английского языка. Тем самым, мы получим порядка 100 миллионов изображений рукописного текста, что значительно превышает размер натурального датасета.

Данный подход имеет свои преимущества и недостатки. С одной стороны, мы имеем огромное количество данных и классов, что может улучшить качество модели при обучении. С другой стороны, человеческий почерк по своей природе представляет из себя более сложную структуру, так как человек пишет каждый раз одну и ту же букву в слове немного по разному, в то время как в шрифтах каждая буква будет абсолютно одинаковой. Это может негативно сказаться на качестве обучения.

2.4 Уменьшение размерности

В архитектуре ResNet, которая используется в качестве бэкбоуна практически во всех моделях обучения в данной работе, выходной слой выдает эмбеддинг размерности 512. Проблема заключается в том, что вектора большой размерности довольно плохо поддаются кластеризации. Главной причиной такого явления является феномен *проклятия размерности*. Его можно интерпретировать разными способами. Одной из формулировок является следующей: пусть n векторов из пространства $\mathbb{R}^d$ распределены в соответствии с некоторым фиксированным распределением. Тогда разница между минимальным и максимальным расстоянием от какой-то фиксированной точки Q до данных точек практически не сравнима с минимальным расстоянием [источник]. Строго говоря:

$$\lim_{d \to +\infty} \mathbb{E} \left(\frac{\text{dist}_{\text{max}}(d) - \text{dist}_{\text{min}}(d)}{\text{dist}_{\text{min}}(d)} \right) = 0$$

Многие алгоритмы кластеризации используют метрики, которые, в следствии вышеописанного феномена, являются довольно слабыми в многомерных пространствах. В связи с этим есть необходимость в уменьшении размерности пространства эмбеддингов.

Задача уменьшения размерности заключается в том, чтобы построить преобразование, которое переводить пространство векторов в пространство наименьшей размерности, при этом добиваясь наименьших потерь. Определение потерь может быть индивидуально для каждого алгоритма.

Для этого существует несколько подходов. Одним из них является классический метод главных компонент, также известный как PCA [источник]. Путем сингулярного разложения матрицы данных, определения главных компонент и проецирования данных на гиперплоскость ортогонально некоторым собственным векторам можно добиться уменьшения размерности с минимальными потерями ковариации. Однако в следствие линейности

данного метода, он не всегда дает хорошего результата при значительном уменьшении размерности.

Другим известным алгоритмом является t-SNE. Представляя из себя нелинейный алгоритм уменьшения размерности, он себя хорошо показывает в задаче визуализации многомерных данных. Однако из-за своего устройства он не способен сохранять расстояния между эмбеддингами и также может создавать искусственные кластера. Это хорошо демонстрируют эксперименты его применения на двух облаках точек, взятые из двух независимых гауссовских распределений [источник].

UMAP является более универсальным алгоритмом уменьшения размерности перед применением алгоритма кластеризации [источник]. Он способен выдавать многообразие меньшей размерности с определенной топологической структурой, если выполняется три условия теоремы [источник]:

1. Данные равномерно распределены на многообразии Римана
2. Метрика Римана локально константна
3. Многообразие локально связно

Данные условия практически всегда выполняются для реальных данных, и, в связи с этим, данный алгоритм хорошо подходит для уменьшения размерности. Также он предоставляет широкий набор параметров, который позволит улучшить качество кластеризации на данных меньшей размерности. Более того, алгоритму можно подать данные с метками, на которых алгоритм может обучиться и выдать более кластеризумые эмбеддинги [источник].

2.5 Кластеризация

Кластеризация является важной частью данной работы. Получив на входе набор эмбеддингов, которые нам дала нейронная сеть и алгоритм уменьшения размерностей, нам нужно теперь выделить кластера рукописных текстов, которые написал один автор.

Существует огромное количество различных алгоритмов, решающих эту задачу. Каждый из алгоритмов по-своему уникален, имеет собственные параметры, учитывает природу данных также по-разному.

Более того, довольно важно учитывать саму природу данных, которые мы пытаемся кластеризовать. Принимая в расчет архитектуры, которые были описаны ранее в данной работе, можно прийти к выводу, что как минимум природа эмбеддингов различается в

метрике, которая минимизировалась для эмбеддингов от одного автора. Как можно вспомнить, например, модель SNN минимизировала евклидово расстояние между двумя векторами, представляющие рукописные тексты от одного писателя. В то же время, модель, обученная на задаче классификации с функцией потерь ArcFace, будет выдавать эмбеддинги, которые расположены на n-мерной гиперсфере и которые обаладют свойством кластеризуемости относительно косинусной метрики.

Из вышесказанного следует, что, выбирая алгоритм, важно учитывать специфику данных. На них очень сильно влияет как архитектура нейронной сети, так и выбранный алгоритм уменьшения размерности.

Хочется отметить два самых главных параметра, которые будут выбираться по-разному в зависимости от выбранной архитектуры модели. Первым является количество кластеров, которое некоторые алгоритмы кластеризации требуют укзаать заранее. Однако далеко не всегда мы знаем их количество. Более того, в данной работе определение количества авторов является одной из поставленных задач. Тем не менее, существует несколько методик определения количества кластеров, которые будут описаны позже. Вторым важным параметром является непосредственно метрика. Некоторые алгоритмы считают эмбеддинги близкими друг к другу именно благодаря метрике, что является логичным утверждением. Она также будет варьироваться в зависимости от выбранной модели.

2.5.1 Алгоритмы кластеризации

Рассмотрим существующие популярные алгоритмы кластеризации.

K-Means

Метод K-Средних является одним из классических алгоритмов кластеризации. Основная задача данного алгортма заключается в минимизации так называемой *инерции*, что также называется *критерием суммы квадратов внутри кластера*:

$$\sum_{k=0}^{n} \min_{\mu_j \in C} (||x_i - \mu_j||^2)$$

Данный подход имеет как преимущества, так и недостатки. Из преимуществ можно выделить широкий положительный опыт его использования на различных типах данных, а также масштабираемость данного метода. Он показывает хорошую производительность на большом количестве данных, существует версия MiniBatchKMeans, которая позволяет порциями

подавать данные для кластеризации.

Однако, так называемая инерция не является оптимальной метрикой. Как минимум, оно требует нормированности пространства эмбеддингов, так как ”растянутые” кластера будут являться контрпримером работы данного алгоритма. Также, алгоритм требует на вход количество кластеров, что для нас является неизвестной величиной, и, возможно, могут появиться трудности с ее определением. Более того, предполагается, что кластера по своей геометрической структуре является выпуклыми множествами относительно зафиксированной метрики. Но при как при обучении наших эмбеддингов, так и при уменьшении размерности, мы это напрямую никак не можем гарантировать, и метод таким образом может показать плохой результат. Также алгоритм плохо себя показывает на векторах высокой размерности, что обусловленно вышеупомянутым феноменом *проклятия размерности*.

Иерархическая кластеризация

Алгоритмы иерархической кластеризации представляют из себя семейство методов, которые объединяет общая идея кластеризации множества векторов путём последовательного слияния и разделения. В начале алгоритма каждый объект находится в собственном кластере. Далее алгоритм находит два самых близких друг к другу кластера и сливает их во едино. Повторяется данный процесс до тех пор, пока не нашлось требуемое количество кластеров.

Данный алгоритм обладает тремя важными параметрами. Первым и вторым, по аналогии с алгоритмом K-Means, являются количество кластеров и метрика соответственно. Третьим параметром подается способ определения расстояния между кластерами. Существует несколько методов для данной задачи. Пусть U, V – кластера точек, $D(U, V)$ – расстояние между кластерами. Тогда существуют как минимум такие методы [источник]:

1. Метод одиночной связи
$$D(U, V) = \min(\rho(u, v))$$
2. Метод полной связи
$$D(U, V) = \max(\rho(u, v))$$
3. Метод средней связи
$$D(U, V) = \frac{1}{|U| \cdot |V|} \sum_{u \in U} \sum_{v \in V} \rho(u, v)$$

4. Метод Уорда

$$D(U,V) = \frac{|U| \cdot |V|}{|U| + |V|} \rho^2 \left(\sum_{u \in U} \frac{u}{|U|}, \sum_{v \in V} \frac{v}{|V|} \right)$$

У данного алгоритма также есть свои преимущества и недостатки. Хочется отметить, что они прежде всего зависят именно от метода определения расстояния между кластерами. В зависимости от него мы можем простоить контрпримеры множеств, которые алгоритм довольно плохо кластеризует. Например, в большинстве случаев данный метод плохо отделяет растянутые кластера. Также, если ему дать равномерно распределенное множество точек, то в некоторых случаях он попробует разделить это пространство на данное ему количество кластеров, хотя оно не кластеризуемо. Тем не менее, в отличие от K-Means алгоритм иерархической кластеризации основывается на меньшем количестве предположений о природе данных, и, таким образом, может показать лучше результат.

MeanShift

Алгоритм MeanShift, или *сдвига среднего значения* является алгоритмом анализа пространства признаков, который также может использоваться как алгоритм кластеризации пространства эмбеддингов. Является итеративным алгоритмом нахождения местоположения максимумов плотности вероятности, которые в последствии могут оказаться нужными нам кластерами.

Сдвиг среднего значения $shift(x)$ вычисляется по следующей формуле:

$$m(x) = \frac{\sum_{x_i \in N_i(x)} K(x_i - x) x_i}{\sum_{x_i \in N_i(x)} K(x_i - x)}$$

$$shift(x) = m(x) - x$$

Имея какое-то начальное значение x_0, оно итеративно сдвигается по следующей формуле:

$$x_n \leftarrow x_{n-1} + shift(x_{n-1})$$

Преимуществом данного алгоритма является тот факт, что на вход ему не надо подавать количество кластеров – это число само появляется во время работы алгоритма.

2.5.2 Определение количество кластеров

Следующим большим вопросом является определение количества кластеров до запуска самого алгоритма кластеризации. Существует два основных подхода к решению дан-

ной проблемы. Первый и самый простой – выбрать алгоритм кластеризации, который не требует на вход количество кластеров. Примером такого алгоритма являются Meanshift, DBSCAN и другие.

При этом мы хотим использовать и иные алгоритмы кластеризации. Для этого мы будем выполнять поиск по сетке по потенциальным значениям количества кластеров и смотреть на метрики, которые будут говорить о том, насколько хорошо кластеризовалось множество эмбеддингов. Существует несколько метрик, которые помогут нам оценить результат кластеризации.

Silhouette Score

Данная метрика определяется следующим образом. Сначала определим ее для одного сэмпла. Пусть s_1 – среднее расстояние от конкретного сэмпла до всех остальных точек в том же самом кластрее, s_2 – среднее расстояние от текущего сэмпла до всех остальных точек в следующем ближайшем кластере. Тогда Silhouette Score для конкретного сэмпла вычисляется по формуле:

$$Silhouette(x_0) = \frac{s_2 - s_1}{\max(s_1, s_2)}$$

Метрика для всех точек вычисляется как средняя от метрики для каждой точки:

$$Silhouette(X) = \frac{1}{|X|} \sum_{x \in X} Silhouette(x)$$

Silhouette Score принимает значения из $[-1; 1]$, где число ближе к 1 является индикатором хорошей кластеризации, -1 – неверной кластеризации, а 0 – пересекающихся кластеров. Соответственно, при определении количества кластеров мы построим график зависимости данной метрики от количества кластеров, и будем искать такое значение аргумента, при котором Silhouette Score выдает максимальное значение.

Недостатком такой метрики отмечают плохую показательность для невыпуклых кластеров, что непосредственно исходит из определения самого метода.

Calinski-Harabasz Score

Индекс Calinski–Harabasz ялвяется альтернативной метрикой для определения количества кластеров с помощью вышеописанного метода. Он определяется как отношение дисперсий между кластерами и внутри кластеров. Более формально, пусть C_i – кластера, определенные каким-то алгоритмом, c_i – центр кластера C_i, c – центр всего множества эмбеддингов, n – количество кластеров. Тогда W – матрица дисперсий внутри кластеров – определяется следующим образом.

$$W = \sum_{k=1}^{n} \sum_{x \in C_k} (x - c_k)(x - c_k)^T$$

O – межгрупповая дисперсионная матрица, определяемая по следующей формуле:

$$O = \sum_{k=1}^{n} |C_k|(c_k - c)(c_k - c)^T$$

Наконец, индекс определяется так:

$$\text{CHScore}(X) = \frac{\text{tr}(O)}{\text{tr}(W)} \times \frac{|X| - n}{n - 1}$$

Данная метрика ведет себя похожим образом, как и Silhouette Score. Чем больше ее значение тем лучше была произведена кластеризация. Среди недостатков можно выделить тот же факт, что она лучше себя показывается именно на выпуклых кластерах.

3 Результаты проведения экспериментов

3.1 Метрики качества

3.2 Эксперименты

4 Заключение