МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Теоретической и прикладной информатики
(полное название кафедры)

УТВЕРЖДАЮ

Зав. кафедрой Чубич В.М.
(фамилия, имя, отчество)

_____
(подпись)

«_____»_____ 2025 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Мусаткина Ильи Дмитриевича
(фамилия, имя, отчество студента – автора работы)

Адаптация статистических методов тематического моделирования для анализа корпусов текстов научных журналов
(тема работы)

Факультет Прикладной математики и информатики
(полное название факультета)

Направление подготовки 01.03.02. Прикладная математика и информатика
(код и наименование направления подготовки бакалавра)

Руководитель от НГТУ	**Автор выпускной квалификационной работы**
Тимофеева А.Ю. (фамилия, имя, отчество)	Мусаткин И.Д. (фамилия, И.О.)
к.э.н., доцент (ученая степень, ученое звание)	ФПМИ, ПМ-12 (факультет, группа)
09.06.2025 (подпись, дата)	**09.06.2025** (подпись, дата)

Новосибирск, 2025 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Теоретической и прикладной информатики
(полное название кафедры)

УТВЕРЖДАЮ

Зав. кафедрой Чубич В.М.
(фамилия, имя, отчество)

«17» марта 2025 г

(подпись)

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

студенту ***Мусаткину Илье Дмитриевичу***
(фамилия, имя, отчество студента)

Направление подготовки *01.03.02. Прикладная математика и информатика*

Факультет Прикладной математики и информатики

Тема ***Адаптация статистических методов тематического моделирования для анализа корпусов текстов научных журналов***

Исходные данные (или цель работы):

целью работы является адаптация вероятностных методов тематического моделирования и оценочных метрик качества моделей для анализа корпуса научных текстов в области Digital Humanities с последующей интерпретацией тематических структур и очерчиванием предметной области

Структурные части работы:

Введение

1. *Теоретические основы тематического моделирования*
2. *Описание программной реализации и используемых библиотек*
3. *Исследования методов тематического моделирования для корпуса журналов*
4. *Интерпретация результатов тематического моделирования*

Заключение

Список литературы

Приложение

Задание согласовано и принято к исполнению.

Руководитель от НГТУ	**Студент**
Тимофеева А.Ю.	*Мусаткин И.Д.*
(фамилия, имя, отчество)	(фамилия, имя, отчество)
к.э.н., доцент	*ФПМИ, ПМ-12*
(ученая степень, ученое звание)	(факультет, группа)
17.03.2025 г.	*17.03.2025 г.*
(подпись, дата)	(подпись, дата)

Тема утверждена приказом по НГТУ № *1479/2* от «*17*» *марта 2025* г.

ВКР сдана в ГЭК № 2.1, тема сверена с данными приказа

17 марта 2025 г.
(подпись секретаря экзаменационной комиссии по защите ВКР, дата)

Целебровская М.Ю.
(фамилия, имя, отчество секретаря экзаменационной комиссии по защите ВКР)

АННОТАЦИЯ

Отчёт 74 с., 4 ч., 12 рис., 4 табл., 51 источн., 1 прил.

ТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ, ВЕРОЯТНОСТНЫЕ ТЕМАТИЧЕСКИЕ МОДЕЛИ, МЕТРИКИ ОЦЕНКИ КАЧЕСТВА МОДЕЛЕЙ, КОРПУС ТЕКСТОВ

Объектом исследования являются методы статистического анализа текстов на естественном языке.

Предмет исследования – возможности применения вероятностных методов тематического моделирования для выявления и интерпретации тем из набора текстов академической специфики.

Цель работы – адаптировать вероятностные методы тематического моделирования и оценочные метрики качества моделей для анализа корпуса научных текстов в области *Digital Humanities (DH)* с последующей интерпретацией тематических структур и очерчиванием предметной области.

В процессе работы разрабатывались программные модули для сбора, предобработки и тематического моделирования текстов, визуализации результатов, расчёта метрик когерентности и тематического разнообразия, а затем производилась интерпретация полученных результатов на основе выделенных тем для слов и статей.

В результате исследования выделены тематики, охватывающие ключевые направления анализируемых журналов научной области *DH*, проведено сравнение моделей (*LDA*, *hLDA*, *DTM*, *ARTM*) и инструментов (*Tomotopy*, *Gensim*, *BigARTM*) тематического моделирования, выявлены зависимости между параметрами моделирования и интерпретируемостью тем, подтверждена эффективность тематического моделирования как средства анализа научных публикаций.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Данная работа посвящена исследованию и адаптации методов статистического тематического моделирования для анализа корпуса текстов, составленного из статей научных журналов в области *Digital Humanities (DH, цифровых гуманитарных наук)*.

Актуальность темы исследования обусловлена тем, что в настоящее время гуманитарные науки всё в большей степени опирается на методы анализа больших текстовых данных, что делает особенно востребованными инструменты *обработки естественного языка (NLP)*, в частности – *тематическое моделирование* (*Topic Modeling, TM*). Методы *TM* зарекомендовали себя как эффективный способ выявления скрытых тем в неструктурированных текстовых данных, и широко применяются в таких областях, как информационный поиск, социология, политическая наука, цифровая история и др.

В области *DH* – сравнительно молодой и междисциплинарной сфере, объединяющей гуманитарные методы и цифровые технологии – *тематическое моделирование* применялось в отдельных проектах [1], однако систематический анализ крупных корпусов статей из профильных научных журналов данной области с использованием адаптированных моделей и альтернативных подходов к *TM* до настоящего времени не проводился.

Особую значимость приобретает задача адаптации *TM* к академическим текстам, характерным для цифровой гуманитаристики. В таких текстах тематические структуры, как правило, более сложные, а терминология нередко полисемантична и варьируется в зависимости от дисциплинарного контекста. Структура большинства документов подчинена логике академического письма, с характерными синтаксическими повторениями и дискурсивными шаблонами.

Настоящее исследование стремится восполнить этот пробел, апробировать различные подходы к тематическому моделированию и

определить множество исследовательских направлений и субдисциплин, формирующих границы *DH* как развивающейся научной области.

Объектом исследования являются методы статистического анализа текстов на естественном языке.

Предмет исследования – возможности применения вероятностных методов тематического моделирования для выявления и интерпретации тем из набора текстов академической специфики.

Цель работы – адаптировать вероятностные методы тематического моделирования и оценочные метрики качества моделей для анализа корпуса научных текстов в области *Digital Humanities* с последующей интерпретацией тематических структур и очерчиванием предметной области.

Для достижения цели исследования ставятся следующие *задачи*.

1. Сформировать корпус научных текстов по тематике *DH* и подготовить его для машинного анализа, учитывая особенности академического жанра.
2. Изучить и проанализировать существующие подходы к тематическому моделированию и их программные реализации с последующим выбором тех модификаций, которые будут использованы для построения и сравнения моделей.
3. Разработать модуль подготовки данных, обучения моделей и извлечения матриц с распределениями слов по темам и тем по документам.
4. Реализовать единое вычисление метрик качества тематического моделирования для всех исследуемых библиотек тематического моделирования.

5. Провести серию экспериментов по исследованию влияния параметров модели (число тем, гиперпараметры, вид предобработки) на итоговое качество и интерпретируемость тем.
6. Выполнить интерпретацию выделенных тем и осуществить попытку определения основных тематик области *DH*, фигурирующих в анализируемом корпусе научных журналов.

Эмпирическим материалом исследования послужили статьи из пяти научных журналов специализации *DH* (проанализировано 1458 статей).

1 ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ

1.1 ЗАДАЧА ТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ

Задача вероятностного тематического моделирования заключается в выделении скрытых семантических структур в коллекции текстовых документов. Основная цель состоит в том, чтобы по заданному корпусу текстов восстановить два распределения:

– распределение тем в каждом документе (матрица Θ);

– распределение слов в каждой теме (матрица Ф).

Документы при этом рассматриваются в упрощённой форме: как мешки слов (*bag-of-words*), то есть без учёта порядка следования лексем. Тематика документа определяется только частотностью вхождения терминов. При таком подходе задача сводится к вычислению низкорангового неотрицательного матричного разложения по вероятностному критерию, обычно максимизации логарифма правдоподобия [2].

Тем не менее, подобная задача в общем случае не имеет единственного устойчивого решения. Согласно критериям корректной постановки задачи по Ж. Адамару, она считается такой, если

1. решение существует;
2. оно единственно;
3. оно непрерывно зависит от входных данных.

Задача тематического моделирования нарушает второй и третий критерии: она обладает множеством допустимых решений из-за латентной природы тем, и демонстрирует высокую чувствительность к изменению параметров модели, ввиду предобработки и конфигурации корпуса. Это означает, что каждая полученная модель представляет лишь одно из возможных приближений к тематической структуре корпуса и требует дополнительной валидации и интерпретации.

Чтобы ограничить пространство решений, в тематические модели вводят априорные распределения или добавляют регуляризаторы. Таким образом, формируется новая целевая функция – модифицированная логарифмом правдоподобия и дополнительными условиями, отражающими лингвистические или структурные предпочтения исследователя.

Таким образом, задача тематического моделирования связана с множеством исследовательских решений, каждое из которых влияет на характер извлекаемых тем. На качество результатов существенно воздействуют:

- *предобработка текста* (фильтрация, лемматизация, удаление имён, частеречная фильтрация и т.д.);
- *выбор числа тем, выбор модели для реализации тематического моделирования и задание гиперпараметров*;
- *выбор и настройка регуляризаторов в случае их применения;*
- *установка порога отсеивания частотных и низкочастотных слов;*
- *интерпретация результатов тематического моделирования.*

Все эти решения влияют на стабильность, интерпретируемость и прикладную пригодность используемой модели.

Основные понятия, используемые в исследовании:

Тематическая модель – это как метод анализа текстов, так и результат его применения: с одной стороны, это статистический алгоритм, выявляющий скрытые темы в корпусе, а с другой – полученное распределение тем по документам и слов по темам (результаты тематической модели).

Метрики – количественные характеристики, с помощью которых измеряется и оценивается модель. Понятие используется в областях компьютерной лингвистики и обработки естественного языка, методы которых применяются и в данной работе. Синоним – *мера*.

Тема (topic) – скрытое распределение вероятностей по термам, отражающее латентную семантику корпуса.

Предобработка – совокупность лингвистических и статистических операций над текстом перед обучением модели.

Токен / терм / слово – в зависимости от этапа обработки используется соответствующее понятие; терм – общее обозначение единицы текста, с которой работает модель, которое пришло из области информационного поиска (*informational retrieval*), из которой берут начало первые методы и проблемы тематического моделирования. Токен – так принято называть единицу обработки в NLP.

Словарь (vocabulary) — множество всех уникальных термов корпуса.

Bag-of-Words (Мешок слов, BoW) – представление документа как *мультимножества слов* (учитывает частоту слов, в отличие от обычного множества слов) без учета порядка. В таком виде подаются корпуса на вход моделям тематического моделирования.

Документ – отдельная текстовая единица в корпусе (например, статья).

Корпус текстов / текстовая коллекция – совокупность документов, на которых обучается модель.

1.2 МОДЕЛИ ВЕРОЯТНОСТНОГО ТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ

Теоретической основой тематического моделирования являются вероятностные модели, в которых используются скрытые (латентные) переменные для описания тематической структуры текстов. Тематическое моделирование исторически возникло как развитие *LSA (Latent Semantic Analysis, латентного семантического анализа)*, основанного на матричной факторизации (например, сингулярном разложении). Однако в отличие от *LSA*, тематические модели нового поколения, такие как *pLSA (Probabilistic Latent Semantic Analysis, вероятностный латентный семантический анализ)* [3] и

LDA (Latent Dirichlet Allocation, латентное размещение дирихле) [4], основываются на генеративной вероятностной модели: предполагается, что каждое слово в документе сгенерировано некоторой скрытой темой, выбранной из распределения тем для этого документа. Такая формулировка позволяет применять байесовский вывод для оценки параметров модели.

Модели *pLSA* и *LDA*, несмотря на общую вероятностную схему, различаются по способу регуляризации. В модели *pLSA* априорные распределения отсутствуют, что делает модель чувствительной к переобучению и неспособной обобщаться на новые документы. В отличие от неё, *LDA* вводит априорные распределения Дирихле, что позволяет учитывать предварительные предположения о разреженности распределений слов и тем [5].

1.2.1 Модель вероятностного латентного семантического анализа (pLSA)

Модель PLSA, предложенная Т. Хофманном в 1999 году, стала первой *вероятностной* моделью, предназначенной для тематического анализа. Она моделирует документы как смеси тем, а темы – как распределения слов. Ключевая идея заключается в том, что каждое слово в документе связано не с самим документом напрямую, а с одной из тем, характерных для этого документа.

Вероятность слова в тексте задаётся для модели как:

$$P(w \mid d) = \sum_{z \in K} P(z \mid d)\, P(w \mid z),$$

где d – документ из коллекции, w – одно из слов словаря, z – одна из скрытых тем, K – *множество скрытых тем,* $P(z \mid d)$ – распределение тем z в документе d, $P(w \mid z)$ – распределение слов w в документе z.

Оценка параметров модели осуществляется с помощью максимизации логарифма правдоподобия по всей коллекции документов, на основе *EM алгоритма (Expectation-Maximization)*.

$$\mathcal{L} = \sum_{d \in M} \sum_{w \in V} n(w, d) * \log \left(\sum_{z=1}^{K} P(z \mid d) P(w \mid z)\right),$$

где $n(w,d)$ – частота слова в документе d, M – множество документов, V – множество слов в словаре (уникальных слов).

С учётом того, что искомые матрицы Φ и Θ являются вероятностными распределениями – их элементы принимают значения от 0 до 1 и в сумме по соответствующему измерению равны единице. Поэтому при оптимизации накладываются два ограничения: неотрицательность их значений (Φ, $\Theta \geq 0$) и нормировка строк (или столбцов) до единицы. Оптимизация модели в этом случае может быть сформулирована как задача минимизации *KL-дивергенции (Kullback–Leibler divergence, дивергенция Кульбака-Лейблера)*, измеряющей расхождение между эмпирическим распределением частот слов в документах F и его приближением через произведение матриц:

$$\|F - \Phi\Theta\|_{KL} \to \min_{\Phi\Theta},$$

где F – эмпирическая матрица частот слов в документах.

Ключевые особенности pLSA:

- вероятностная интерпретация – в отличие от классической LSA, которая может содержать отрицательные значения, модель работает с вероятностными неотрицательными распределениями, что повышает интерпретируемость результатов [3];
- использование латентных переменных – тема z выступает как скрытая причина появления слова в документе, что делает модель способной выявлять структурные закономерности в больших корпусах;
- отсутствие априорных распределений – модель строится без дополнительных предположений о виде распределений параметров, что отличает её от *LDA*.

Ограничения модели:

– необобщаемость на новые документы – распределение тем задаётся отдельно для каждого текста обучающей выборки, поэтому модель не может применяться к новым данным без повторного обучения;
– отсутствие регуляризации – pLSA не ограничивает множество возможных решений, что делает модель чувствительной к переобучению и неустойчивой. В терминах Адамара, задача остаётся некорректной: решение не единственно и нестабильно. Регуляризация, напротив, позволяет вводить предпочтения или ограничения на параметры модели (например, разреженность), тем самым приближая задачу к корректной постановке и делая вывод более управляемым.

1.2.2 Модель латентного размещения Дирихле (LDA)

LDA – вероятностная модель тематического моделирования, предложенная в 2003 году Д. Блеем, Э. Ыном и М. Джорданом. Она основывается на модели *pLSA*, устраняя её ключевые ограничения. В *LDA* вводятся априорные распределения Дирихле на скрытые переменные, что позволяет осуществлять полнобайесовский вывод – при котором выводятся не просто оценки параметров, но полное апостериорное распределение всех скрытых переменных и параметров. Это даёт устойчивость к переобучению и способность модели к порождению «псевдодокументов» на основе распределения.

Вероятностные допущения, для которых предполагается априорное подчинение матриц Φ и Θ *распределению Дирихле*:

– распределение тем по документу: $\theta_d \sim Dir(\alpha)$, θ_d – вектор вероятностей размерности *K,* в сумме равный единице;
– распределение слов по теме: $\varphi_z \sim Dir(\beta)$, φ_z – вектор вероятностей размерности *V,* в сумме равный единице.

Таким образом, θ_d отражает предпочтения документа d к темам, а φ_z – тематический профиль темы z, то есть вероятностное распределение слов, характерных для данной темы.

Формула полной вероятности для LDA:

$$P(w, z, \theta, \varphi \mid \alpha, \beta) = \prod_{d}^{M} P(\theta_d \mid \alpha) \prod_{z}^{K} P(\varphi_z \mid \beta) \prod_{t}^{N_d} P\left(z_{d,t} \mid \theta_d\right) P\left(w_{d,t} \mid \varphi_{z_{d,t}}\right),$$

где

- N_d – число слов в документе d, $d = \overline{1, M}$,
- α – параметр априорного распределения Дирихле для распределения тем в документах,
- β – параметр априорного распределения Дирихле для распределения слов по темам,
- $z_{d,t}$ – тема, к которой отнесено j-е слово в i-м документе,
- $w_{d,t}$ – конкретное j-е слово в i-м документе,
- $\varphi_{z_{d,t}}$ – распределение слов в теме $z_{d,t}$.

На рисунке 1 представлено, как в общем виде выглядит схема *LDA*:

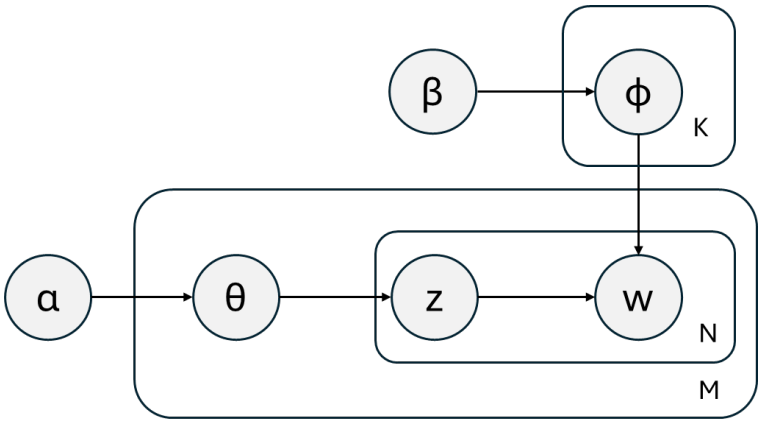

Рисунок 1 – Табличная нотация LDA

Низкие значения гиперпараметра α способствуют тому, что распределения тем в документах становятся разреженными – документ характеризуется небольшим числом выраженных тем. Аналогично, низкие значения β приводят

к тому, что каждая тема описывается компактным набором слов с высокой вероятностью.

На практике для обучения модели *LDA* применяются два основных метода: *VB (Variational Bayes, вариационный байесовский вывод)* и *семплирование Гиббса (Collapsed Gibbs Sampling)*.

Одним из направлений критики модели *LDA* является отсутствие лингвистически обоснованной интерпретации для априорных распределений Дирихле. Эти распределения используются в первую очередь для удобства математического вывода и обеспечения разреженности, но они не учитывают специфики языковых феноменов [5].

1.2.3 Модели надстройки над LDA

Современные задачи тематического моделирования часто требуют учета временной динамики и иерархической структуры тем. Для решения этих задач разработаны модели, расширяющие классическую *LDA*.

1. *DTM (Dynamic Topic Model) [6]*.

Позволяет моделировать эволюцию тем во времени. В отличие от *LDA*, которая предполагает, что порядок документов не имеет значения, *DTM* учитывает временную последовательность документов, группируя их по временным срезам (например, по годам). Предполагается, что темы в каждом временном срезе эволюционируют из тем предыдущего среза, что позволяет отслеживать изменения в тематике документов со временем.

Вероятность слова $w_{d,t}$ из документа *d*, относящегося ко времени *t*, задаётся как

$$P\left(w_{d,t}\middle|\theta_{d,t},\varphi_{z,t}\right)=\sum_{z}P(z\mid\theta_{d,t})\,P\left(w_{d,t}\mid\varphi_{z,t}\right),$$

где $\theta_{d,t}$ – распределение тем в документе на временном шаге *t,* $\varphi_{z,t}$ – распределение слов в теме *z* на временном шаге *t.*

Для темы z в каждом временном срезе *t:* определяется распределение слов $\varphi_{z,t}$, зависящее от $\varphi_{z,t-1}$ для каждой темы *z*. Для каждого документа *d* в срезе *t*: выбирается распределение тем $\theta_{d,t}$, зависящее от $\theta_{d,t-1}$.

2. hLDA (Hierarchical Latent Dirichlet Allocation) [7].

hLDA моделирует темы в виде иерархии, что позволяет выявлять как общие, так и специфические темы в корпусе.

Модель строится на основе *nCRP (nested Chinese Restaurant Process, вложенный китайский ресторанный процесс)*, стохастической процедуры, порождающей дерево тем с гибкой, адаптивной структурой: каждый документ выбирает путь в дереве, вероятностно продолжая уже существующие ветви или создавая новые. При этом вероятность выбора пути зависит от того, сколько других документов уже выбрали этот путь – по аналогии с тем, как посетители ресторана выбирают стол.

Иерархия тем определяется параметром *depth,* определяющим максимальное количество уровней дерева (обычно 2 или 3), а также гиперпараметром *gamma,* стандартно задаваемым равным *0,1*. Этот параметр регулирует вероятность порождения новых ветвей: меньшие значения способствуют формированию более компактной иерархии, тогда как большие – увеличивают разнообразие тем.

1.2.4 Модель аддитивной регуляризации тематических моделей (ARTM)

Аддитивная регуляризация тематических моделей (Additive Regularization for Topic Modeling, ARTM) – это современный подход к построению тематических моделей, разработанный российской исследовательской группой под руководством К. Воронцова. *ARTM* представляет собой обобщение модели *pLSA*, к которой добавлена возможность включать регуляризаторы, формализующие дополнительные предпосылки о тематической структуре.

Модель решает задачу максимизации логарифма правдоподобия, аналогично *pLSA*, но с добавлением регуляризационных членов:

$$\mathcal{L}(\Phi, \Theta) = \log P(W \mid \Phi, \Theta) \; + \; \sum_i \tau_i \, R_i(\Phi, \Theta),$$

где i – индекс используемых в модели регуляризаторов, каждый из которых вносит вклад в показатели модели, R_i – регуляризаторы, τ_i – коэффициенты регуляризации.

Таким образом, *ARTM* устраняет ключевые ограничения *pLSA*, такие как некорректная постановка задачи по Адамару (отсутствие единственности и устойчивости решения), и предлагает альтернативу *LDA*, избегая использования лингвистически необоснованных априорных распределений.

Регуляризаторы позволяют исследователю целенаправленно управлять интерпретируемостью модели. Например, высокая разреженность матрицы Φ позволяет выделить более отчётливые темы, а снижение корреляции между темами помогает избежать их избыточного пересечения. В данной работе будут применены *регуляризаторы разреженности матриц* Φ и Θ, а также *декоррелизатор матрицы* Φ.

1.3 КОЛИЧЕСТВЕННЫЕ МЕТРИКИ ДЛЯ ОЦЕНИВАНИЯ МОДЕЛЕЙ ТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ

В тематическом моделировании, как и в других областях машинного обучения, оценка качества моделей является ключевым этапом, особенно в условиях отсутствия разметки данных. В данном исследовании акцент сделан на использовании частотных статистических метрик, поскольку они обеспечивают объективную и воспроизводимую оценку качества тематических моделей, особенно при невозможности проведения исследования интерпретируемости с помощью экспертной группы людей.

1.3.1 Перплексия

Перплексия (perplexity) является одной из ключевых метрик оценки качества тематических моделей, таких как LDA. Она измеряет, насколько

хорошо модель предсказывает распределение слов в документах, и интерпретируется как мера “запутанности” модели: чем ниже перплексия, тем лучше модель описывает данные.

Перплексия вычисляется по формуле:

$$PP = \exp(-\frac{1}{N}\sum_{d=1}^{D}\sum_{w\in d} n_{dw} \log P(w \mid d)),$$

где n_{dw}- число вхождений слова *w* в документ *d,* $P(w \mid d) = \sum_{k=1}^{K} \theta_{dk}\varphi_{kw}$.

Для каждого документа вычисляется логарифм вероятности слов, взвешенный по их частотам, и затем нормализуется на общее количество слов в корпусе.

1.3.2 Метрики когерентности

Метрики *когерентности* оценивают степень тематической связности слов внутри темы. Они основаны на статистике совместной встречаемости пар слов из топ-k термов темы в корпусе. При этом сами значения когерентности рассчитываются точечно (*pointwise*) для каждой пары слов (w_i, w_j) из k, после чего агрегируются (в данном исследовании было использовано усреднение по темам) – сначала внутри темы, затем по всем темам.

1. U_{Mass}

Метрика U_{Mass} вычисляет, как часто два слова совместно встречаются в корпусе. Глобальная когерентность темы считается как средняя точечная когерентность по топ *k* словам, описывающим тему.

$$U_{Mass} = \frac{1}{K}\sum_{t=1}^{K}\left(\frac{1}{\binom{k}{2}}\sum_{i=1}^{k}\sum_{j=1}^{i-1}\log\left(\frac{D(w_i w_j)+\varepsilon}{D(w_i)}\right)\right),$$

где $D(w_i, w_j)$ – частота, с которой пара слов встречается вместе, $D(w_i)$ – частота, с которой слово встречается отдельно от второго, ε – сглаживающий коэффициент чтобы избежать неопределённости логарифма, в оригинальной статье $\varepsilon = 1$ [8].

U_{Mass} доказала эффективность для коротких текстов, где локальная контекстная информация критична. Однако, метрика может быть чувствительна к шуму в данных, особенно в корпусах с высокой лексической вариативностью. Метрика принимает значения в диапазоне $[-1,0]$.

2. C_{UCI}

Основывается на скользящих окнах и *PMI* – точечной взаимной информации - всех пар слов [9]. Соответственно, встречаемость для пары слов считается не в рамках документа, но в рамках одного окна фиксированной длины. Если метрика использует внешний корпус для расчёта, то она зависит от его содержания и может не подходить для анализа корпуса научных журналов, но в рамках ручной реализации, использующейся в данной работе, а также в реализации в библиотеке *tomotopy* метрика считается на том же корпусе, на котором обучалась модель.

$$PMI = \frac{P(w_i, w_j)}{P(w_i)P(w_j)},$$

$$\mathrm{C}_{UCI}(w_i, w_j) = \log\frac{P(w_i, w_j) + 1}{P(w_i)\ P(w_j)},$$

где $P(w_i, w_j)$ – вероятность появления пары слов в пределах заданного окна.

3. NPMI

NPMI [10] нормализует *PMI* в диапазоне $[-1,1]$. Значения $>$ *0,5* считаются отличными, *0,2–0,5* – приемлемыми, $<$ *0,2* – слабыми. Для собственной реализации будем использовать модификацию с фиксированным окном по всему документу.

$$NPMI(w_i, w_j) = \frac{PMI(w_i, w_j)}{-\log\ P(w_i, w_j)}.$$

4. C_v

C_v [11] создаёт вектора слов (векторные эмбеддинги, например, с помощью *Word2Vec*) на основе их совместной встречаемости, после чего вычисляет рейтинг используя *NPMI* и косинусное сходство между векторами. Конечное значение нормализуется до диапазона [0, 1] с использованием набора значений когерентности и их среднего значения [Там же], где значения > 0,6 считаются высокими.

Так выглядит использующаяся в работе косинусная мера связанности тем, вычисляющаяся для пары слов:

$$\cos(w_i, w_j) = \frac{v(w_i) * v(w_j)}{\|v(w_i)\| * \|v(w_j)\|},$$

где $v(w_i)$ – вектор совместной встречаемости слова w_i с другими словами в корпусе.

Средняя когеретность темы по всем парам слов *P* в теме:

$$Coh(W_t) = \frac{1}{P} \sum_{w_i, w_j \in P} \cos(w_i, w_j).$$

Итоговая когерентность C_v как среднее по всем темам:

$$\mathrm{C}_v = \frac{1}{T} \sum_{t \in T} Coh(W_t).$$

Исследователи отмечают [5], что *UMass* и *UCI* могут лучше работают на структурированных корпусах (новости, научные статьи), тогда как *NPMI* и C_v адаптивны к коротким и зашумленным текстам. Однако, последние имеют большую корреляцию с человеческими оценками.

1.3.3 Разреженность и энтропия

Разреженность матриц Φ и Θ отражает, насколько каждый термин или документ ассоциирован с ограниченным числом тем. Она рассчитывается как доля элементов, меньших заданного порога числа ε, от общего числа значений в матрице:

$$Sparsity = \frac{\text{количество значений} < \varepsilon}{\text{общее количество значений в матрице}}.$$

Высокая *разреженность* матрицы Ф указывает на маленькое число терминов с высокой вероятностью, что может свидетельствовать о качественной кластеризации слов по темам [5] и формировании устойчивого семантического ядра.

Энтропия, напротив, отражает степень равномерности распределения термов в теме. Она вычисляется *по формуле Шеннона*:

$$Entropy = -\sum w\varphi_{kw} \log(\varphi_{kw}).$$

Низкая *энтропия* означает, что тема фокусируется на небольшом числе терминов (то есть распределение концентрировано), тогда как Высокие значения энтропии характерны для слабо выраженных или размытых тематик. Эти характеристики дополняют метрики *когерентности* и *перплексию*, позволяя судить о внутренней структуре модели.

1.3.4 Метрики тематического разнообразия

- *Тематическое разнообразие (TD)*

Это метрика, оценивающая степень уникальности ключевых слов в темах, полученных с помощью тематического моделирования. Она основана на концепции лексического разнообразия, но адаптирована для анализа распределения слов между темами, а не внутри отдельного текста.

$$TD = \frac{\text{Число уникальных слов в топ } k \text{ слов всех тем}}{\text{Общее число слов в топ } k \text{ слов всех тем}}.$$

Высокий показатель меры *TD* (*Topic Diversity, тематическое разнообразие*) зачастую свидетельствует о хорошей дифференциации тем, но не стоит считать его абсолютным показателем качества модели, поскольку сама

задача тематического моделирования предполагает разумное пересечение слов между темами – ведь вероятности слов распределены между несколькими темами. Это отличает задачу тематического моделирования, от, например, задач жёсткой кластеризации. *TD* отражает лишь степень уникальности топовых слов и не учитывает их семантическую связность или содержательную значимость, поэтому интерпретировать эту метрику следует в комплексе с другими показателями, такими как тематическая когерентность и экспертная оценка осмысленности тем.

- *RBO (RBD)*

RBO (Rank-Biased Overlap)[12] – это ещё одна метрика тематического разнообразия, которая оценивает уникальность тем на основе ранжированных списков ключевых слов. Для каждой темы t выбирается топ-k терминов с наибольшими значениями вероятности $p(w \mid t)$. Попарное сходство между темами вычисляется с помощью инвертированной и модифицированной меры *RBD (Inverted RBO)* адаптированной для анализа тематического разнообразия. *RBO* измеряет степень перекрытия между двумя ранжированными списками с учётом позиции термов и параметра затухания $p \in (0,1)$, задающего важность верхних позиций.

Стандартная формула *RBO*:

$$RBO = (1-p)\sum_{d=1}^{k} \frac{A_d \cap B_d}{d} * p^{d-1},$$

где A_d и B_d – множества первых d терминов в соответствующих списках.

Модифицированная мера:

$$RBD_{norm} = 1 - \frac{RBO}{RBO_{max}}, RBO = \sum_{d=1}^{k} \frac{A_d \cap B_d}{d} * p^{d-1},$$

где RBO_{max} – теоретически максимальное значение RBO при полном совпадении списков. Полученная метрика принимает значения от *0* до *1*, где *0*

соответствует полному совпадению ключевых слов тем (отсутствие различий), а *1* – их полному различию (максимальная уникальность).

1.4 ВЫВОДЫ

В данной главе были рассмотрены основные концептуальные и математические основы тематического моделирования текстов. Обозначена постановка задачи и её вероятностная формализация, связанная с факторизацией матрицы частот и введением скрытых переменных, интерпретируемых как темы. Подчёркнута некорректная постановка задачи в смысле Адамара, что требует введения дополнительных ограничений – априорных распределений (в *LDA*) или регуляризаторов (в *ARTM*).

Были проанализированы ключевые тематические модели: *pLSA* – как базовая модель без регуляризации; *LDA* – как обобщение с байесовскими предпосылками; а также их надстройки: *DTM* для учёта временной динамики и *hLDA* для иерархического представления тем. Кроме того, отдельное внимание уделено подходу *ARTM*, предполагающему гибкую конфигурацию модели через систему регуляризаторов.

Особое внимание уделено метрикам оценки качества моделей: перплексии как вероятностному показателю пригодности модели к данным, а также семейству метрик когерентности, отражающих интерпретируемость тем через частотные связи между словами. Рассмотрены дополнительные показатели, такие как разреженность, энтропия, тематическое разнообразие и ранговая дистанция, применяемые в задачах сравнения и отбора моделей.

Таким образом, глава формирует теоретическую основу для дальнейшей части исследования, посвящённой практической реализации, сравнительному анализу моделей и интерпретации их результатов на корпусе научных текстов в области *Digital Humanities*.

2 ОПИСАНИЕ ПРОГРАММНОЙ РЕАЛИЗАЦИИ И ИСПОЛЬЗУЕМЫХ БИБЛИОТЕК

2.1 ОПИСАНИЕ СТРУКТУРЫ РАЗРАБОТАННЫХ ПРОГРАММНЫХ МОДУЛЕЙ

Для реализации всех модулей программы использовался язык *python* и совместимые с ним библиотеки. Такой выбор обуславливается тем, что данное исследование затрагивает множество разделов: веб-скрапинг, парсинг, работа со строками, методы *NLP*, методы тематического моделирования, работу с матрицами, вычисление статистических метрик и построение визуализаций. Каждый из этих разделов можно реализовать с помощью *python*. Для лучшей воспроизводимости исследования на других устройствах было принято решение о реализации не через *.py*, а чере *.ipynb* файлы. Все зависимости задокументированы в файл *requirements.txt* для упрощения развёртывания

Таким образом, разработанные программные блоки реализованы в виде *Jupyter-ноутбуков*, каждый из которых отвечает за отдельный этап анализа: сбор и предобработка корпуса, обучение моделей, вычисление метрик качества и визуализация результатов. От архитектуры программной части проекта требовалась модульность и последовательность. Основная цель при проектировании состояла в том, чтобы сбалансировать строгость структуры и гибкость использования: с одной стороны, избегать избыточности и дублирования, с другой – обеспечить удобство настройки параметров и возможность расширения под различные сценарии исследования без необходимости переработки всей системы. Структурно программные модули разделены на

- модуль предобработки текстов и подсчета их статистик,
- основной модуль тематического моделирования,
- модуль тематического моделирования расширенными методами,

- модуль тематического моделирования для проведения кластеризации по журналам.

2.2 ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ ПРОГРАММНЫХ МОДУЛЕЙ

Выбор инструментов обусловлен необходимостью решения задач различного уровня: от веб-скрапинга и предобработки текстов до построения тематических моделей, анализа их качества и визуализации результатов.

– *Обработка данных и файловые операции*

pandas, numpy – обеспечивают удобные структуры данных и операции для табличных представлений и числовых вычислений.

json, pathlib – используются для сериализации данных и кроссплатформенной работы с файловой системой.

collections, itertools – оптимизированные контейнеры и генераторы для частотного анализа и комбинаторных операций.

– *Веб-скрапинг и парсинг*

requests, BeautifulSoup – позволяют извлекать и разбирать HTML-контент с сайтов научных журналов;

langdetect – применяется для автоматического определения языка и фильтрации нерелевантных текстов.

– *Обработка текста и NLP*

spaCy – основная библиотека для токенизации, лемматизации и распознавания именованных сущностей (*NER*).

nltk – дополнительные инструменты для работы с корпусами и стоп-словами.

– *Тематическое моделирование*

gensim – реализация итерационного процесса *LDA* основана на *онлайн-аппроксимации вариационного вывода (Online Variational Bayes)*. Используется метод *gensim.models.LdaModel*, принимающий на вход *BoW-корпус* и словарь *dictionary.*

tomotopy – модель реализована на основе *семплирования Гиббса (Collapsed Gibbs Sampling)*, отличается высокой скоростью и устойчивостью. Кроме того, содержит большое число имплементаций методов на основе *LDA*, расширяющих функционал метода.

Сравнение реализаций *LDA* представлено в таблице 1.

Таблица 1 – Сравнение LDA в *tomotopy* и *gensim*

Gensim	Tomotopy
gensim.models.LdaModel (corpus, num_topics=10, id2word=dictionary, passes=10, alpha='symmetric', eta=None)	*class LDAModel (tw=TermWeight.ONE, min_cf=0, min_df=0, rm_top=0, k=10, alpha=0.1, eta=0.01, seed=None, corpus=None, transform=None)*
– *id2word* – словарь *gensim*, сопоставляющий id слов с самими словами; – *passes* – количество проходов по всему корпусу при обучении	– *tw* – схема взвешивания; – *min_cf, min_df* – пороги частот; – *rm_top* – игнорировать наиболее частые слова; – *seed* – “зерно” для инициализации генератора случайных чисел; – *transform* - метод трансформации входного корпуса перед обучением.

Общими для моделей являются 1) *“alpha”* и *“eta”* – гиперпараметры распределения Дирихле α и β (В tomotopy значения *“alpha”* и *“eta”* всегда задаются явно в виде скаляров, в отличие от gensim, где возможны как симметричные, так и асимметричные распределения); 2) *“corpus”* – корпус в виде *BoW*; *“num_topics”(“k”)* – число тем извлекаемых моделью.

ARTM – библиотека для реализации моделей тематического моделирования с регуляризаторами.

Используемые в исследовании регуляризаторы:

- *artm.SmoothSparsePhiRegularizer(tau=1.0, gamma=None, ...)* – разреживание или сглаживание в матрице Φ (слово-тема);
- *artm.SmoothSparseThetaRegularizer(tau=1.0, alpha_iter=None, ...)* – разреживание или сглаживание в матрице Θ (тема-документ);
- *artm.DecorrelatorPhiRegularizer(tau=1.0, gamma=None, ...)* – уменьшает пересечения между темами.

Поддерживает два метода обучения:

- *model.fit_offline(batches)* – классический EM-подобный алгоритм с многократным проходом по данным. Данный метод используется в данной работе;
- *model.fit_online(batches)* – обновляет параметры по мере поступления батчей (подходит для потокового обучения).

В настоящей работе библиотека *ARTM* используется для построения базовой модели *pLSA* (нулевые регуляризаторы) и для последующего анализа влияния различных регуляризаций на структуру тематик корпуса научных статей.

- *Визуализация и анализ*

matplotlib, *seaborn* – построение статистических графиков.

pyLDAvis – библиотека для python, производящая интерактивную визуализацию распределения тем [13]. Каждая тема представлена вектором распределения вероятностей слов в этой теме. Сходство между темами оценивается на основе этих векторов, и затем применяется метод многомерного шкалирования (*MDS*) для отображения тем в двумерном пространстве. Таким образом, темы, находящиеся ближе друг к другу на визуализации, имеют более похожие распределения слов, что указывает на их тематическую близость

2.3 ВЫВОДЫ

В рамках данной главы была рассмотрена разработанная универсальная программная система, позволяющая автоматизировать весь цикл тематического моделирования: от сбора и предобработки текстов до интерпретации результатов. Модульная структура реализованных компонентов позволяет повторно использовать и адаптировать отдельные этапы для различных корпусов и задач.

Преимуществом реализации стало сохранение прозрачности всех этапов: пользователь может контролировать параметры моделей, отслеживать ход итераций, настраивать визуализации, адаптировать модуль метрик под конкретную задачу.

Основные программные модули и фрагменты кода приведены в *Приложении* и могут быть воспроизведены в любом *python*-совместимом окружении.

Данная система может быть использована не только в рамках анализа научных текстов, но и в других исследовательских задачах, где требуется выявление скрытых тем в корпусе текстов.

3. ИССЛЕДОВАНИЕ МЕТОДОВ ТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ ДЛЯ КОРПУСА ЖУРНАЛОВ

3.1 ПОДГОТОВИТЕЛЬНЫЙ ЭТАП ИССЛЕДОВАНИЯ

3.1.1 Сбор статей

Для проведения исследования научной области *DH* были найдены основные журналы, со статьями на английском или преимущественно на английском языке, посвященные этой области знания. Кроме того, было значимо, чтобы журналы не были посвящены какой-либо одной подобласти цифровых гуманитарных наук, как, например, журнал *“Digital Medievalist”* (посвящённый применению цифровых технологий для изучения средневековья) и продолжают выпускаться в настоящее время (поэтому не был взят журнал *“Journal of Digital Humanities”*). Итого было отобрано пять журналов: *“Digital Humanities Quaterly”* (далее *DHQ*), *“Digital Scholarship in the Humanities”* (далее *DSH*), *“Digital Studies / Le Champ Numérique”* (далее *LChN*), *“Journal of Cultural Analytics”* (далее *JCA*) и *“International Journal of Digital Humanities”* (далее *IJDH*).

Отобранные сайты отличались по числу представленных статей, а также степенью доступности к контенту бесплатно: от полного доступа ко всем статьям на сайте *DHQ* до доступа лишь к малой доле статей на сайте *IJDH.*

Для сбора текстов открытых статей с сайтов журнала был разработан программный модуль, использующий веб-скрапинг и парсинг данных в *json*-файлы, создавая коллекцию текстов с основным полем *“text”*, куда сохранялся извлечённый текст статьи без заголовков и списка литературы, а также с полями для метаданных *“url”, “title”, “year”, “journal”*. Первые два поля хранят в себе ссылку на страницу статьи на сайте и её название для возможности более удобно верифицировать результаты тематического моделирования. В третьем поле хранится год выпуска статьи для возможности

дальнейшей реализации модификации алгоритма тематического моделирования с применением временной кластеризации. Поле *“journal”* используется для хранения аббревиатуры журнала, к которому принадлежит статья для возможности проведения межжурнального сравнения.

Итоговая статистика по отобранным корпусам представлена в таблице 2.

Таблица 2 – Итоговая статистика по отобранным корпусам

	Весь корпус	DHQ	DSH	LChN	JCA	IJDH
Количество статей	1 458	739	231	302	146	40
Количество слов	9 302 789	4 545 710	1 419 092	1 938 695	1 140 487	258 805
Среднее количество слов на документ	6 380	6151	6 143	6 419	7 811	6 470

Хотя корпусы естественным образом и отличаются по количеству статей, средняя количество слов на документ во всех журналах примерно одинаково. Это позволяет применять единые подходы к моделированию журналов, а также объединить их в один корпус для анализа, поскольку случай корпусов коротких текстов, например, аннотаций статей или публикаций в социальных сетях (*50 – 300* слов), требует особого подхода и адаптации моделей.

3.1.2 Предобработка корпуса статей

Для полученных текстовых коллекций, готовых к подаче на вход тематическим моделям, был реализован программный модуль предобработки текстовых коллекций. Используемые библиотеки описаны *в разделе 2*.

Для начала все слова были приведены к нижнему регистру и удалены все не *ASCII*-символы, пунктуация, цифры и лишние пробелы с помощью библиотеки *re.* Файл предобработанного корпуса обозначается постфиксом *“_T”*.

Дальнейшие инструменты предобработки могут применятся от пожеланий исследователя. Из предобработанного корпуса могут быть удалены имена собственные и произведена лемматизация (постфикс *"_L"*) – приведение слова к его базовой словарной форме (лемме). Также возможно использование частеречной разметки (*POS-тэгирование*) с помощью библиотеки *SpaCy*, использующей скрытовую марковскую модель. По результатам частеречной разметки в корпусе остаются только слова указанных частей речи. Далее может производиться удаление из корпуса токенов, встречающихся в сформированном для данного исследования словаре стоп слов. (постфикс *"_S")*

Адаптированный для работы с научными журналами словарь стоп слов включает в себя не только частовстречающиеся общие и служебные конструкции английского языка. Для дополнения словаря частотными малоинформативными словами получили список *500* самых частотных униграмм (одиночных слов) и *100 биграмм* (пар слов) по корпусу. Из этого списка были вручную отобраны частовстречающиеся специфические для академического дискурса на английском языке слова, такие как *define, describe, object, focus, provide, refer, explore* и другие. Всего было добавлено около *80* слов, после извлечения которых объём словаря объединённого корпуса журналов уменьшился почти в *2* раза: с *9 302 789 до 5 000 360* термов.

Частеречную разметку в рамках исследования разделили на 3 варианта: 1) все части речи (*"_ALL"*); 2) существительные и глаголы (*"_NOUNVERB"*); 3) только существительные (*"_NOUN"*). Такое решение было принято исходя из того, что, отыскивая скрытые темы, нужно концентрироваться на поиске значимых для текстов слов, а больше всего смысловой информации содержат, как правило, существительные и глаголы.

Итого получилось 24 варианта предобработки исходного корпуса, влияние которых на результат моделирования будет обсуждаться ниже.

3.2 ВЫБОР БИБЛИОТЕК, МОДЕЛЕЙ И ВАРИАНТОВ ПРЕДОБРАБОТКИ ДЛЯ ТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ

3.2.1 Влияние предобработки на значения метрик

Наибольшее влияние на результаты уже выбранной модели исследователь оказывает двумя вещами: выбором числа тем и методами предобработки исходной коллекции текстов. Первый аспект будет рассмотрен позднее. Остановимся на влиянии предобработок и попытаемся определить лучший из вариантов на основе результатов метрик, а в качестве значений параметров модели будем использовать стандартные показатели библиотеки *tomotopy*.

Проведённое на нескольких корпусах исследование показало, что при удалении имен собственных из текста статей результаты незначительно улучшаются по большинству метрик. Поэтому, в дальнейшем будем использовать корпуса только с удаленными именами собственными, а в таблице приведём результаты замеров уже для *12* текстов, сократив обзор вдвое.

В таблице 3 приведены значения метрик когерентностей, перплексии, тематического разнообразия (*TD*) и *RBD*, использована реализация *LDA* в *tomotopy*.

Можно увидеть, что все показатели когерентности, кроме U_{Mass} выделяют в качестве двух наиболее связных корпусов *TSL_ALL* и *TS_NOUN*. Сравнивая варианты *POS-тэгирования*, на большей части корпусов лучше всего значения у корпусов только из существительных, а хуже всего – у коллекций со всеми частями речи. Однако эта корреляция перестаёт быть явной, если рассматривать корпуса, из которых исключены стоп-слова. Можно заметить, что наибольшее влияние на показатель связности моделей оказывает влияние именно извлечение *стоп-слов*, которое основано в данном иследовании на словаре, специально подготовленном под специфику научных статей.

Таблица 3 – Сравнение методов предобработки

Тип предобработки	*TD*	*RBD*	*Perplexity*	U_{Mass}	C_{UCI}	$NPMI$	C_v
T_ALL_N	0.273	0.202	1291	-0.160	-0.011	-0.01	0.384
T_NOUNVERB_N	0.791	0.950	3619	-0.483	0.407	0.063	0.557
T_NOUN_N	0.809	0.951	2282	-0.793	0.245	0.075	0.594
TL_ALL_N	0.264	0.234	919	-0.155	-0.003	-0.009	0.390
TL_NOUNVERB_N	0.818	0.958	1886	-0.548	0.447	0.075	0.571
TL_NOUN_N	0.818	0.976	1483	-0.548	0.447	0.075	0.571
TS_ALL_N	0.746	0.876	5175	-0.533	0.405	0.061	0.554
TS_NOUNVERB_N	0.855	0.985	4032	-0.616	0.503	0.078	0.580
TS_NOUN_N	0.873	0.978	2534	-0.744	0.538	0.091	0.609
TSL_ALL_N	0.864	0.969	3710	-0.506	0.572	0.087	0.61
TSL_NOUNVERB_N	0.809	0.980	2206	-0.534	0.354	0.065	0.559
TSL_NOUN_N	0.836	0.965	1666	-0.571	0.423	0.074	0.563

Лемматизация играет меньшую роль на изменение метрик, но её применение к корпусу *TS_ALL* позволило ему стать лучшим по большинству показателей. Это происходит за счёт того, что лемматизация снижает лексическое разнообразие корпуса с максимальным значением перплексии среди всех корпусов.

За счёт этого снижается и перплексия и модель начинает лучше предсказывать данные. Однако, перплексия, хотя и является одной и самых известных метрик в контексте тематического моделирования, не может использоваться как единственный показатель для оценки модели - опираясь лишь на неё, можно сделать вывод о том, что наилучшая версия предобработки произведена для корпуса *TL_ALL*. Этот корпус действительно имеет наименьший показатель “запутанности” для предсказания, ведь в нём оставленны все наиболее частвстречающиеся в текстах служебные слова, например, *“the”, “of”, “that”,* а кроме того, к нему применена лемматизация, в результате чего лексическое разнообразие было уменьшено и такие

частовстречающиеся слова как, например, *“are”, “is”* были на общее слово *“be”* с суммированным значением частоты. Кроме того, высокая перплексия может свидетельствовать о недостаточной обученности модели.

Однако, один из показателей когерентности - U_{Mass} показывает обратные по отношению к остальным метрикам результаты. Эта единственная из рассматриваемых метрик когерентности, которая учитывает совстречаемость (co-occurence) слов не по окну заданной длины, но по всему документу. Из-за этого уменьшение размера словаря в результате продвинутых предобработок не улучшает значение метрики. Оставшиеся показатели улучшаются, поскольку позволяет «приблизить» друг к другу важные слова, удалив малоинформативные, находившиеся между ними, позволяя им чаще попадать в «окно» совстречаемости. Шумные слова слишком влияют на показатели модели, поэтому на неё, как и на перплексию нельзя смотреть в отрыве от контекста типа анализируемого корпуса.

Метрики тематического разнообразия (*TD* и *RBD*) подтверждают результаты оценки с помощью когерентностей C_V, C_{UCI} и $NPMI$, при этом являясь менее вычислительно затратными. Если высокие значения когерентности интерпретируются как высокая связность между топ-словами темы, то высокие значения *TD* и *RBD* указывают на наибольшую несхожесть между топ-словами разных тем, говоря о хорошем показателе разреженности.

Можно заметить низкое влияние от включения / исключения глаголов в состав корпуса. Это можно связать, во-первых, с тем, что в текстах научных статей глаголы не обладают такой семантической значимостью, как, например, в текстах художественной литературы, в которых развитие сюжета связано с действиями героев, которые выражаются с помощью глаголов, ведь текст статьи «статичен». Во-вторых, разнообразие используемых в статьях глаголов невелико и значимая их часть была включена в адаптированный стоп-словарь.

В связи с этим, в дальнейших исследованиях откажемся от использования версий текстов формата “*существительное + глагол*”, а также от предобработок не использующих стоп-слова. Для дальнейшего исследования предлагается остановиться на 2 лучших вариантах предобработки – 1) с применением стоп-слов и лемматизации с сохранением всех частей речи (*TSL_ALL*) и 2) с применением стоп-слов к корпусу только из существительных (*TL_NOUN*).

Аналогичное исследование было проведено с использованием реализации gensim и на основе ручного вычисления метрик когерентности. Полученные корреляции совпадают с результатами модели *LDA tomotopy*.

На рисунке 2 представлены визуализации тем, полученных на двух вариантах предобработки корпуса. Слева – базовая токенизация (*T_ALL*) (худший вариант), справа – предобработка с выделением только существительных и стоп-слов (*TS_NOUN*) (один из лучших вариантов). Круги обозначают темы, их размер – удельную долю темы в корпусе, а расстояние между кругами – различие между темами по распределению слов. Можно наблюдать, что в версии *TS_NOUN* темы более чётко разделены, расположены дальше друг от друга, и представлены более равномерно, тогда как в *T_ALL* наблюдается перекрытие тем и явное доминирование одной. Это демонстрирует влияние предобработки на интерпретируемость и устойчивость тематической структуры.

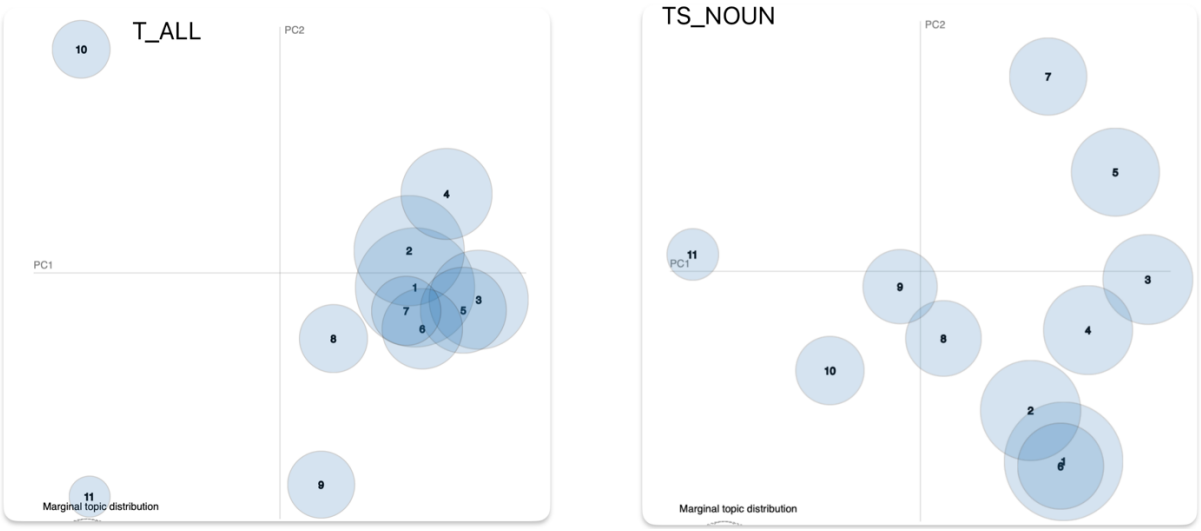

Рисунок 2 – Сравнительная визуализация распределения тем

3.2.2 Влияние на результаты модели размеров окон, использующихся для вычисления метрик когерентности

Данная метрика не зависит от общего размера коллекции документов, но чувствительна к длине отдельных текстов. При слишком малом размере окна модель может не зафиксировать устойчивые связи между терминами – особенно в научных текстах, где ключевые понятия могут быть разделены малозначимыми словами. С другой стороны, чрезмерно большое окно может включать нерелевантные термины, что приводит к увеличению шума в расчётах.

В практике тематического моделирования для корпусных исследований чаще всего используют *окна длиной 10–20 слов* при вычислении метрик C_{UCI}, а также *100–120 слов* – для C_v. Эти значения эмпирически подобраны, в частности, в работе [11], как компромисс между точностью и устойчивостью оценки когерентности.

Для корпуса *TSL_ALL* было проведено моделирование с различными значениями размера окна. Цель эксперимента заключалась в том, чтобы определить, при каких параметрах метрики когерентности позволяют наиболее уверенно выделить корпус с качественной тематической структурой.

Таблица 4 – Влияние размера окна на метрики когерентности

Размер окна	10	**20**	30	40	60
C_{UCI}	0.6879	**0.7120**	0.7028	0.6856	0.65
Размер окна	80	110	120	**140**	160
C_v	0.6677	0.6691	0.6692	**0.6694**	0.6692

Для дальнейших экспериментов *были зафиксированы размеры окон равные 20 и 140.*

3.2.3 Сравнение библиотек для тематического моделирования методом LDA

Для сравнения качества моделей *LDA*, реализованных в *python*-библиотеках *gensim* и *tomotopy*, был проведён ряд экспериментов на корпусе *TSL_ALL*. Основная цель заключалась в сопоставлении результатов по ключевым метрикам: разреженности и энтропии матриц Φ и Θ, тематическому разнообразию, метрике *RBD*, перплексии, а также метрикам когерентности (U_{Mass}, C_{UCI}, $NPMI$, C_v).

В обоих случаях число итераций было зафиксировано на уровне *100*. Однако стоит отметить, что в *gensim* используется дополнительный параметр passes, задающий количество полных проходов по корпусу (эпох обучения), тогда как в Tomotopy обучение по умолчанию выполняется по итерациям (обновление параметров после каждого шага). Даже при одинаковом количестве итераций и одном проходе (*passes=1*) время обучения модели *gensim* оказалось примерно в 10 раз выше по сравнению с *tomotopy*.

По большинству основных параметров – *разреженность и энтропия матрицы Φ, энтропия матрицы Θ, тематическое разнообразие, RBD, перплексия, метрики когерентности (кроме U_{Mass})* - модель *LDA gensim* показала худшие показатели. Единственная метрика, по которой преимущество наблюдалось у модели *gensim* – U_{Mass}, однако она зависит от частотных данных корпуса и плохо коррелирует с интерпретируемостью тем.

Повышение значения passes до *50–100* позволяет модели *gensim* приблизиться или частично обогнать *tomotopy* по отдельным метрикам, но сопровождается резким увеличением времени обучения, вплоть до 100 раз.

С другой стороны, увеличение числа итераций в *tomotopy* выше 100 не дало заметных улучшений качества – более того, по отдельным метрикам (например, когерентности и разнообразию) наблюдалось ухудшение результатов, что можно интерпретировать как эффект переобучения. Поэтому, для дальнейших исследований было принято значение ***iterations = 100.***

Учитывая более высокое качество тематических моделей, меньшую чувствительность к параметрам и значительно более высокую скорость обучения, для последующих экспериментов методом *LDA* в данной работе была использована библиотека *tomotopy*. Её дополнительными преимуществами являются встроенные реализации расширений базовой модели *LDA*, что делает её более гибкой для исследований.

3.2.4 Сравнение LDA с моделью на основе pLSA с применением аддитивной регуляризации

Модель *ARTM* отказывается от байесовской предпосылки о том, что распределения по темам и словам порождаются из априорных распределений Дирихле, тем самым возвращаясь к вероятностной парадигме *pLSA*. В отличие от *LDA*, *ARTM* поддерживает гибкий механизм управления структурой модели с помощью регуляризаторов. В рамках данного исследования были использованы три из них: *artm.SmoothSparsePhiRegularizer, artm.SmoothSparseThetaRegularizer, artm.DecorrelatorPhiRegularizer.*

Для оценки влияния регуляризации на качество тематической модели, результаты сравнивались с базовой моделью *ARTM* без регуляризаторов, которая, по сути, эквивалентна классической *pLSA*.

На рисунке 3 представлены результаты изменения перплексии в зависимости от количества тем $k = \overline{10,20}$ для моделей *pLSA* и *ARTM*.

Поскольку встроенная в *tomotopy* функция вычисления перплексии даёт значения, завышенные в *2,5–3* раза, для всех моделей использовалась самостоятельно разработанная реализация метрики, соответствующая теоретическому описанию *в главе 1*.

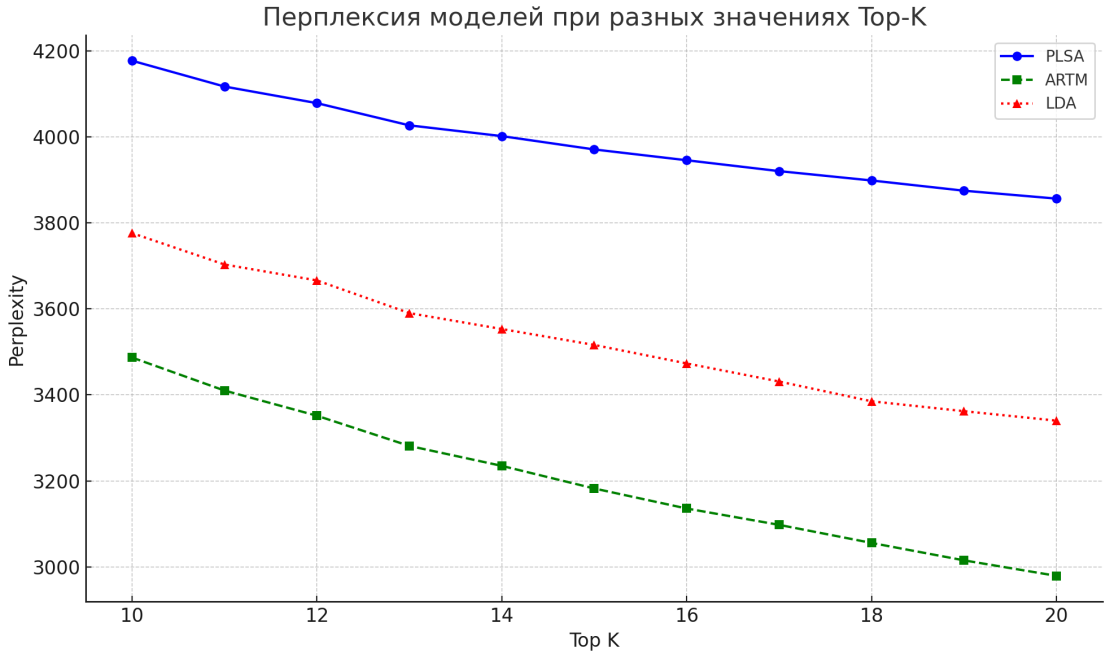

Рисунок 3 – Перплексия моделей при разном количестве тем

Из рисунка видно, что для всех моделей значения перплексии планомерно снижаются на промежутке *от 10 до 20* тем. Снижение продолжается и при большем числе тем, но постепенно его скорость снижается. Модель *ARTM* существенно улучшает перплексию по сравнению с моделью без регуляризаторов *pLSA* и показывает лучшие значения, чем *LDA* при большем времени работы.

Разреженность, определяемая как доля нулевых элементов в матрицах Φ и Θ, плавно возрастает с увеличением числа тем в *ARTM* за счёт работы регуляризаторов. В отличие от неё, модели *LDA* и *pLSA* формируют плотные матрицы: они не обнуляют вероятности, даже если значения малы, и потому разреженность равна нулю. Таким образом, матрицы *ARTM* содержат большое число нулей, что уменьшает объём хранимых данных примерно в 1,5 раза при исследуемом диапазоне тем. Однако это не делает распределения менее

«размазанными»: в *LDA* большинство значений пусть и не являются нулями, но определяются очень малыми (*e-8*) числами, которые не вносят вклада в тему.

Следовательно, несмотря на то, что темы, выделяемые *ARTM*, структурно отличаются от тем *LDA*, качественные различия между ними в пределах $k = 10$ не очевидны. Возможно, различия станут более выраженными при значительно большем числе тем.

В целом, *ARTM* представляется перспективной библиотекой с широкими возможностями кастомизации. Однако в рамках данного исследования она не показала существенного прироста качества по сравнению с классическими моделями. Кроме того, в настоящий момент *ARTM* содержит ограниченный набор встроенных метрик оценки и требует более сложной установки, чем стандартные *python*-библиотеки, распространяемые через *pip*. Также поддержка *ARTM* ограничена: библиотека не гарантированно работает на всех операционных системах.

3.3 ИССЛЕДОВАНИЕ ОПТИМАЛЬНЫХ ПАРАМЕТРОВ ДЛЯ ОБУЧЕНИЯ ТЕМАТИЧЕСКИХ МОДЕЛЕЙ

3.3.1 Выбор оптимальных гиперпараметров модели LDA

В рамках экспериментов было проведено исследование влияния гиперпараметров α и β, а также числа тем k на качество тематического моделирования, выполненного на объединённом корпусе статей. Целью эксперимента являлся выбор диапазона и конкретных значений параметров, обеспечивающих наилучший баланс между статистической устойчивостью модели и содержательной интерпретируемостью её результатов.

На практике значения α и β регулируют, соответственно, распределение тем в документах и слов в темах. Малые значения α способствуют тому, что каждый документ содержит ограниченное число тем, тогда как низкие значения β способствуют формированию узко определённых, разреженных тем. В

рассматриваемых библиотеках, принято задавать значения $\alpha \in [0.01, 1.0]$, $\beta \in [0.001, 0.1]$, с последующей калибровкой в зависимости от размеров корпуса и цели анализа.

Для визуального анализа были построены графики изменения перплексии и тематического разнообразия (*topic diversity*) при изменении числа тем от 2 до 50. Верхняя граница диапазона обусловлена, во-первых, ограничениями интерпретируемости: исследователю затруднительно качественно осмыслять более полусотни тем. Во-вторых, с учётом размера корпуса (около 1500 документов), большое количество тем может привести к фрагментации – появлению тем, представленных в единичных текстах, что не соответствует целям данного анализа, направленного на выявление сквозных тем области *DH*.

На рисунке 4 представлены значения метрик при изменении α от 0,01 до 2 и β от 0,0005 до 0,2. Эти значения охватывают как стандартные, так и граничные случаи, позволяя оценить устойчивость модели и чувствительность к параметрам. В дальнейшем, аналогичная процедура будет применена к корпусам отдельных журналов, с целью определения их оптимального числа тем. Для сравнения использовались *встроенная в tomotopy* метрика перплексии и вручную подсчитываемая метрика тематического разнообразия.

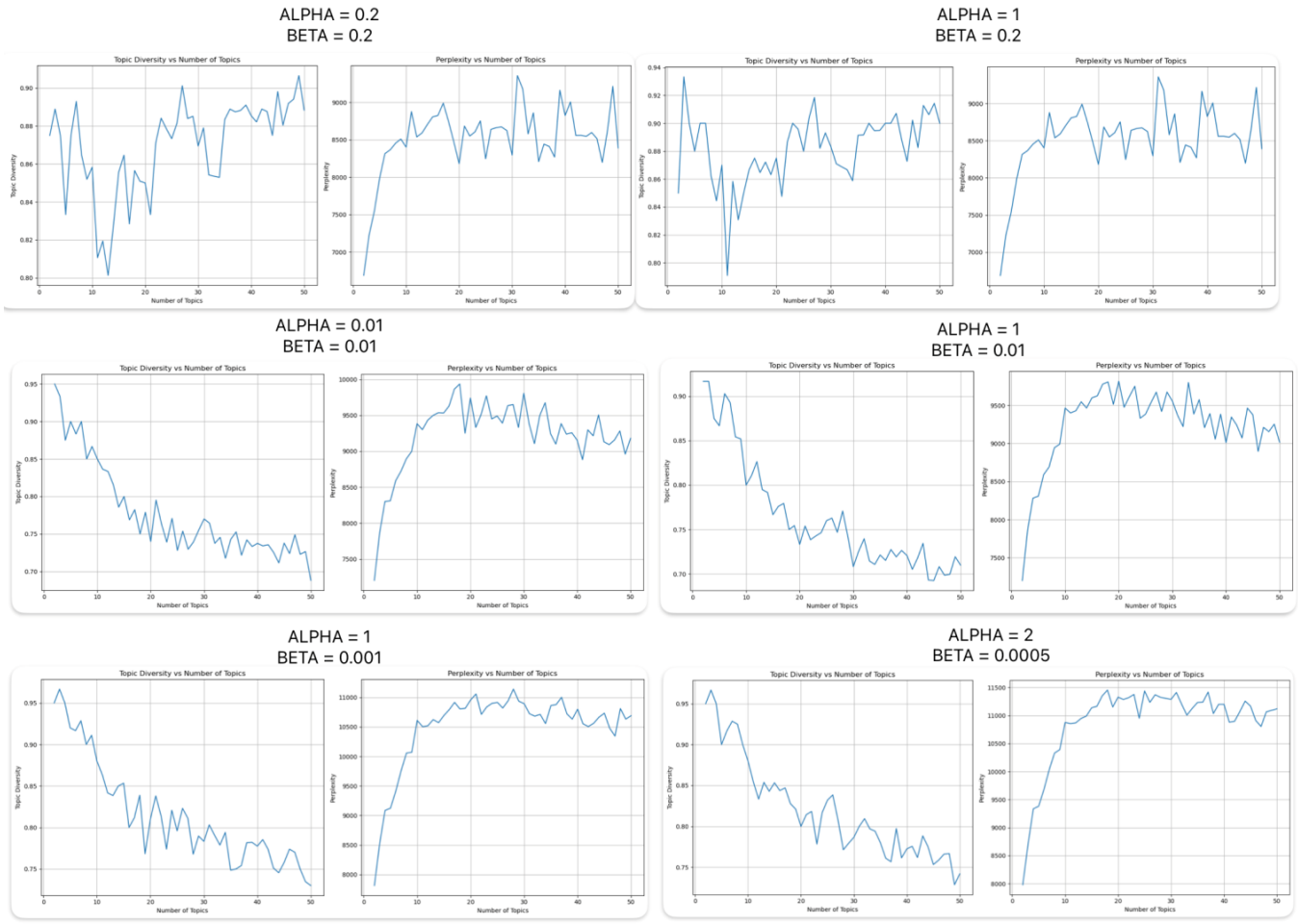

Рисунок 4 – Зависимость перплексии и тематического разнообразия от числа тем при разных значениях гиперпараметров

Для выбора оптимального числа тем малые числа тем не рассматривались, поскольку несмотря на высокие значения метрик они не способны адекватно отобразить тематическую структуру корпуса из 1500 статей.

Перплексия модели, как показатель запутанности предсказания слов в документах, демонстрирует закономерное снижение при увеличении значения гиперпараметра β. Данный эффект объясняется тем, что увеличение β способствует распределению вероятности появления слов в теме по большему числу слов, тем самым и уменьшая «запутанность» модели в предсказывании. В результате, модель становится менее специализированной, что отражается в снижении значения перплексии.

Тем не менее, высокие значения β приводят к значительной нестабильности большинства метрик – в первую очередь тематического разнообразия, а также перплексии. Для оценки влияния β на поведение когерентности были рассмотрены две метрики: $NPMI$ и C_v. При стандартных (0.01) и пониженных (0.001) значениях β обе метрики демонстрируют рост на интервале числа тем от 2 до 14, после чего их значения стабилизируются. Значение $NPMI$ стабилизируются около *0,1*, что интерпретируется как средний уровень связности, а C_v достигает *0,6*, что принято считать высоким качеством тематической структуры.

Таким образом, можно заключить, что увеличение β выше стандартных значений вызывает высокую вариативность во всех показателях, что снижает интерпретируемость и надёжность модели. Вместе с тем, высокие значения β позволяют некоторым моделям с большим числом тем сохранять высокие когерентности и тематическое разнообразие, в отличие от случаев с малыми β, где наблюдается снижение и выход на «плато». Поэтому, низкое значение β предпочтительно ввиду стабильности при изменении тем.

В контексте настоящего исследования было принято решение *зафиксировать значение β на уровне 0,001*. Данный уровень обеспечивает лучшие показатели тематического разнообразия по сравнению со стандартным значением (*0,01*), в то время как *перплексия* пусть и становится выше, но демонстрирует большую устойчивость. При этом дальнейшее снижение β (например, до *0,0005*) не приводит к заметному улучшению ни *тематического разнообразия*, ни *когерентности*, а по *перплексии* наблюдается деградация модели. Кроме того, выбор такого значения параметра соответствует специфики корпуса научных статей, для которого предполагается, что каждая тема, как правило, определяется небольшим числом специфических слов.

Параметр α, регулирующий распределение тем внутри документов, оказывает значительно меньшее влияние на итоговую структуру модели. Тем не менее, в ходе детального анализа удалось выявить положительное влияние

увеличения α на улучшение показателей тематического разнообразия и когерентности, а также отсутствие влияния на перплексию.

Кроме того, выбор осуществлялся исходя из интересов исследования: большие значения α предполагают увеличенное значение числа важных тем в документе, а значит позволят более вероятно учесть большее число тем, а не останавливаться на нескольких частотных темах, пусть возможно и малоинформативных. В связи с этим, в дальнейших экспериментах параметр α зафиксирован на уровне *1*.

3.3.2 Выбор оптимального числа тем для модели LDA

На рисунке 5 представлены изменения значений метрик модели от числа тем для объединённого корпуса.

K	TD	Perplexity	NPMI Coherence	CV Coherence
13	0.8141	10624.5254	0.0718	0.5757
14	**0.8274**	**10572.4057**	**0.0921**	**0.6165**
15	**0.8278**	10694.0148	0.0794	0.5931
16	0.7760	10793.4733	0.0817	0.5934
17	0.7892	10916.6188	0.0717	0.5852
18	**0.8194**	**10804.5630**	0.0787	**0.6017**
19	0.7544	10815.7492	0.0696	0.5705
20	0.7833	10958.8074	0.0723	0.5850

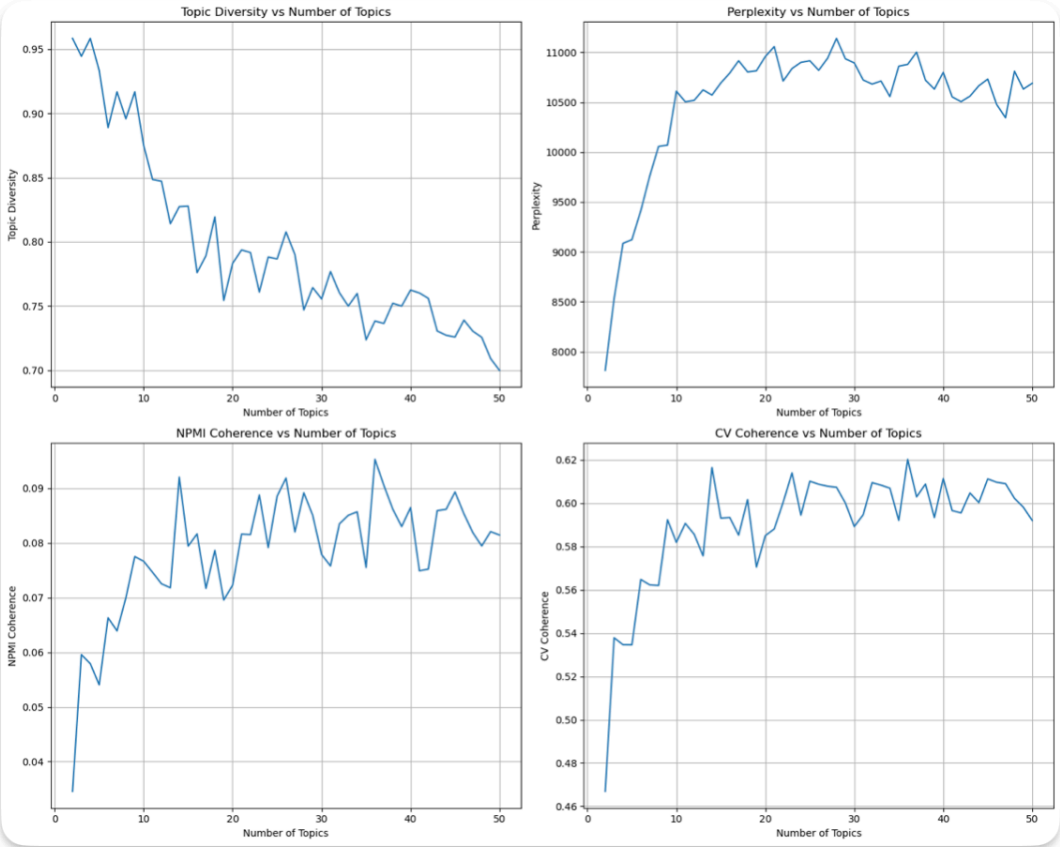

Рисунок 5 – Изменение метрик от числа тем при выбранных значениях гиперпараметров

При *14* темах наблюдаем локальный максимум значений метрик *когерентности*, *тематического разнообразия* (наравне с $k = 15$), а значение *перплексии* при данном значении остаётся умеренным. Такое значения числа тем подойдёт для выделения «широких» тем, при этом оно не является ультимативным – для попытки определить большее число более «узких» тем, можно найти локальные оптимумы при большем числе тем. В качестве альтернативного варианта было зафиксировано $k = 18$, при котором достигается следующий локальный оптимум.

Объединённый корпус был кластеризован по принадлежности статей к журналам, после чего был проведён аналогичный эксперимент, в котором были определены предпочтительные значения тем для каждого журнала:

- для всего корпуса – 14 или 18,
- для *DHQ – 12,*
- для *DSH* – 10,
- для LChN – 11,
- для IJDH – 7.

3.4 ВЫВОДЫ

В ходе практической части исследования был проведён анализ влияния параметров предобработки, гиперпараметров моделей и числа тем на метрики качества. Показано, что библиотека *Tomotopy* обеспечивает лучшее соотношение качества и скорости по сравнению с *Gensim*, а оптимальные значения гиперпараметров существенно влияют на интерпретируемость и стабильность тем. Для объединённого корпуса было подобрано оптимальное число тем и параметры модели.

Исследованы надстройки над классической моделью *LDA*, такие как иерархическая модель (*hLDA*) и динамическая (*DTM*), которые позволяют выделять более узкие и специализированные темы, давая более детальный срез содержательной структуры корпуса.

Библиотека *BigARTM* была протестирована и показала хорошее качество выделения тем и возможность гибкой настройки по сравнению с другими библиотеками, однако обладает ограниченным числом встроенных метрик оценки и поддерживается не на всех операционных системах.

В данной части работы были выбраны оптимальные параметры для тематического моделирования на корпусе журналов, результаты которого детально рассматриваются и интерпретируются *в главе 4.*

4 ИНТЕРПРЕТАЦИЯ РЕЗУЛЬТАТОВ ТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ

4.1 ИНТЕРПРЕТАЦИЯ РЕЗУЛЬТАТОВ МОДЕЛИ LDA

4.1.1 Результаты для объединённого корпуса текстов

Проведём процесс тематического моделирования *LDA* для объединённого корпуса *TSL_ALL*. Среди двух лучших корпусов по итогам первого исследования – *TSL_ALL* и *TL_NOUN* – первый показал лучшие значения по всем метрикам, кроме *перплексии*, при подобранных оптимальных параметрах: число тем – *14*, числом учитываемых топ слов – *10*, при значениях параметров $\alpha = 1$ и $\beta = 0{,}001$.

Работа с методами тематического моделирования предполагает от исследователя на каждом этапе работу по подбору оптимальных значений для количественных метрик модели, учитывая итоговые результаты модели в виде списка топ-слов. Данная задача не является тривиальной и не сводится к лучшему оптимальному решению, поскольку на каком бы значении параметров не остановился исследователь, некоторые темы не будут включены в итоговые результаты модели.

На последнем этапе исследования в библиотеке *tomotopy* исследователь может гибко влиять на результаты тематической модели путём подбора значений для параметров

- *rm_top* – задаёт количество наиболее частотных слов, которые будут исключены из модели (аналог исключения глобальных стоп-слов);
- *min_cf* – минимальное количество вхождений слова во всём корпусе (*collection frequency*). Слова, встречающиеся реже, исключаются из модели;

– *min_df* – минимальное количество документов, в которых должно встретиться слово (document frequency). Слова, не достигшие этого порога, также исключаются.

Такая тонкая настройка позволяет исключить малоинформативные токены, с отсеиванием которых не справилась предобработка и получить варьирующиеся тематические срезы.

Были обучены модели для значений *rm_top = 0, 10, 100* и *min_cf (min_df) = 1,2.* Большая часть тем, пусть и с незначительными изменениями в составе топ слов сохранялась при выводе. Однако, в итоговом списке из 14 выявленных тем, при стандартных значениях *rm_top = 0*, несколько повторяющихся тем были заменены выявленными специфическими темами при альтернативных значениях параметров, для большей репрезентативности конечного списка, представленного на рисунке 6.

Рисунок 6 – Топ-слова в темах по результатам тематического моделирования

Как видно из рисунка 5, полученные темы обладают высокой интерпретируемостью: топ-слова внутри каждой темы демонстрируют семантическую согласованность и формируют чёткие смысловые кластеры. Это позволяет говорить о качественной работе модели и её способности выявлять содержательные тематические структуры.

Ниже представлены краткие интерпретации тем, полученных в результате моделирования, с обращением к топу тем-документов, сформированных моделью на основе матрицы Θ. Это позволило определить, каким образом тематическая структура отражает содержание корпуса.

4.1.2 Анализ и интерпретация выделенных тем

Тема 1. Данная тема охватывает проблематику архивов, баз данных и источников, применяемых в гуманитарных и исторических исследованиях. Характерное сочетание терминов, таких как archive, database, collection, source, указывает на цифровую инфраструктуру, играющую роль как объекта, так и инструмента исследования. В статье [14] подчёркивается двойственная роль исследователя-архивиста в DH-практиках. Упоминание database отражает интерес к нетрадиционным источникам, включая пиратские платформы цифровых текстов [15], а также указывает на разработку и курирование архивов как компонент исследовательской среды [16].

Тема 2. *game, play, narrative, player* – центральные токены, относящиеся к области game studies. Тематика объединяет интерактивные медиа, генерацию повествования [17] и критические обзоры литературы о компьютерных играх [18].

Тема 3. *poem, verse, meter, translation* – слова, фиксирующие поэтические формы, перевод, ритмику, и указывающие на пересечение филологических подходов с цифровой обработкой текстов. Тема охватывает как формы поэтической практики, так и инструменты анализа лингвистических особенностей.

Тема 4 сформировалась в результате высокой внутренней однородности фрагментов на языке оригинала и их низкой связности с остальным корпусом: несмотря на исключение франкоязычных текстов, в корпусе остались англоязычные статьи о французской литературе, включающие многочисленные цитаты на языке оригинала. Эти фрагменты, слабо связанные с основным корпусом, были идентифицированы моделью как отдельная тема [19].

Тема 5. Главное слово в теме - *manuscript (манускрипт)* - необработанный текстовый источник. Работе по комментированию, реконструкции и цифровизации таких источников посвящён достаточно большой пласт работ в сфере цифровой гуманитаристики, например, она затрагивается в [20] и [21]. С

ним связаны термы *TEI (Text Encoding Initiative)* и *XML (Extensible Markup Language*, *Расширяемый язык разметки)*. *TEI* является международным стандартом разметки текстов с использованием *XML*, который применяется и для разметки манускриптов. *textual* отсылает к дисциплине *textual scholarship*, а слова вроде *version, line, name* логично вписываются в контекст разметки и анализа текстов.

Тема 6. *doi, access, press, book, edit, university, library* – токены, связанные с академическим публикационным процессом, распространением научных текстов и инфраструктурой открытого доступа. *DOI* указывает на цифровую идентификацию научных публикаций, что соотносится с институциональной средой – университетами и библиотеками.

Тема 7 объединяет исследования цифровой аудиовизуальной культуры, где звук и музыка анализируются через призму жанров, ритма и алгоритмических методов. Радиопередачи в Швеции и Дании изучаются в рамках нейросетевого анализа речи и музыки [22], [23]. В глобальных исследованиях выявляются закономерности музыкального потребления и жанровые предпочтения [24]. Отдельный проект предлагает алгоритмическое преобразование классического мюзикла для анализа аудиовизуального стиля [25].

Тема 8. Данная тема отличается от других выраженной технической направленностью: в неё входят специфические термины, связанные с вычислениями и цифровой средой, при этом многие из них являются одними из самых частотных в корпусе. Наличие слов kind и thing, относящихся к общеупотребительной лексике, объясняется тем, что они не были исключены на этапе предобработки, а высокая частотность привела к их объединению с более значимыми терминами. В целом, тему можно охарактеризовать как отражающую общее техническое поле цифровой гуманитаристики.

Тема 9. Два ключевых слова темы – eebo и tcp — являются частью аббревиатуры *EEBO-TCP (Early English Books Online Text Creation Partnership)*

– международного проекта, цель которого – создание структурированных транскрипций старых английских печатных книг. С 2015 года результаты проекта стали общедоступны, что может объяснять большое количество исследований, опирающихся на корпус *EEBO*. Выделяемые наиболее вероятные статьи в этой теме в основном посвящены исследованиям классической английской литературы и старого английского языка цифровыми методами. Так, статья [26] посвящена прототипированию цифровой базы английской литературы эпохи Возрождения. Статья [27] исследует пьесы английского драматурга XVI века Томаса Кида с применением как цифровых, так и традиционных методов. В статье [28] производится кластерный анализ религиозных руководств XVI века, что объясняет попадание в топ слова sermon. Слова *king* и *shakespeare* связаны со статьёй [29], посвящённой анализу пьес Шекспира, включая те, что связаны с образом английских королей. Появление слова *anonymous* объясняется статьёй [30], где с применением цифровых интерпретативных инструментов анализируется восприятие творчества Натаниэля Готорна, в том числе через призму анонимных критических откликов XIX века.

Тема 10. Анализ пяти статей, вошедших в данную тему, позволяет заключить, что она посвящена такой подобласти цифровых гуманитарных наук и компьютерного текстового анализа, как *стилометрия*. Так, статья [31] посвящена стилометрии (или компьютерной стилистике), применяемой к диалогам в кино, включая сентимент-анализ и подходы дальнего чтения. В статье [32] методами стилометрии исследуются тексты английской романистки XVII века Афры Бен. Статья [33] посвящена математическому объяснению метрики дельта Берроуза – одной из ключевых в автороведении. Четвёртая по весу в теме статья [34] подробно рассматривает применение метода главных компонент в стилометрии. Наконец, в пятой статье [35] предлагается оригинальный метод стилистического анализа, направленный на устранение источников предсказуемых вариаций. Хотя само название области (*stylometry*)

не фигурирует в топе слов, тематическая модель успешно выделяет и объединяет статьи, посвящённые именно ей.

Тема 11. В данную тему входят статьи, посвящённые академическим институциям, образовательным программам и инфраструктуре цифровых гуманитарных наук. Статья [36] посвящена статья посвящена исследованиям учебных программ бакалавриата и магистратуры направления *DH*. В работе [37] обсуждается роль академических библиотек в развитии этой области и формирования сообщества исследователей, а в [38] рассматривается роль центра ETCL – канадского института цифровых научных практик – в изменении ландшафта дисциплины. Статья [39] описывает опыт открытия и функционирования десяти DH-центров в Северной Америке.

Тема 12 посвящена развитию цифровой филологии и инфраструктуры цифровых библиотек, работающих с текстами на древнегреческом, латинском, а также с текстами Средневековья и эпохи Возрождения. Например, статья [40] рассматривает процессы тестирования и интеграции в таких проектах, как Perseus Digital Library (PDL) и Open Philology Project (OPP), публикующих и поддерживающих подобные текстовые корпуса. Тем самым формируется поле, где филология, цифровая инфраструктура и историко-культурное наследие соединяются в единое исследовательское направление.

Тема 13 преимущественно посвящена практике оцифровки музейных коллекций и работе с культурным наследием в институциональных контекстах. Это подтверждается ключевыми словами *museum* и *heritage*. В частности, такие практики обсуждаются и в статье, посвящённой развитию цифровых гуманитарных исследований в России на базе Сибирского федерального университета[41]. Одной из затрагиваемых тем в статье являются музейные проекты, в том числе национальные, чем объясняется попадание в топ слова токена *national*.

Тема 14 объединяет исследования, посвящённые культурным, этническим и гендерным аспектам, а также тому, как смена эпох влияет на восприятие и

интерпретацию текстов. Например, статья [42] исследует вклад послевоенных британских писательниц в культурное пространство. В работе [43] с помощью сетевого анализа изучаются устойчивые и меняющиеся паттерны чтения и литературной репутации, отражающие влияние современных культурных, гендерных и этнических сдвигов на восприятие модернистской литературы. Исследование [44] также основано на данных книжного магазина *Shakespeare and Company*, где проводится статистический анализ различий в читательских предпочтениях мужчин и женщин. На этой основе делаются выводы о возможностях переосмысления установившихся канонов, открывая поле для формирования альтернативных представлений о литературном поле.

В результате проведённого анализа и интерпретации корпуса удалось выявить ряд устойчивых тематических направлений, отражающих как научные области, объединяемые полем *Digital Humanities*, так и характерные инструменты и подходы цифровых исследований: архивы и базы данных, исследования игр и нарратива, поэзия и перевод, манускрипты и разметка текстов, академическое издание и открытый доступ, цифровая аудиовизуальная культура, техническая терминология *DH,* корпус *EEBO* и старая английская литература, стилометрия и определение авторства текстов, обсуждение институций и образовательных программ *DH*, античная и средневековая цифровая филология, музеи и культурное наследие, исследования человеческой идентичности.

4.1.3 Сравнение результатов тематического моделирования между журналами

Помимо исследования коллекции текстов как единого целого, была применена кластеризация текстов по принадлежности к журналу.

На рисунке 7 представлено распределение выявленных тем по журналам.

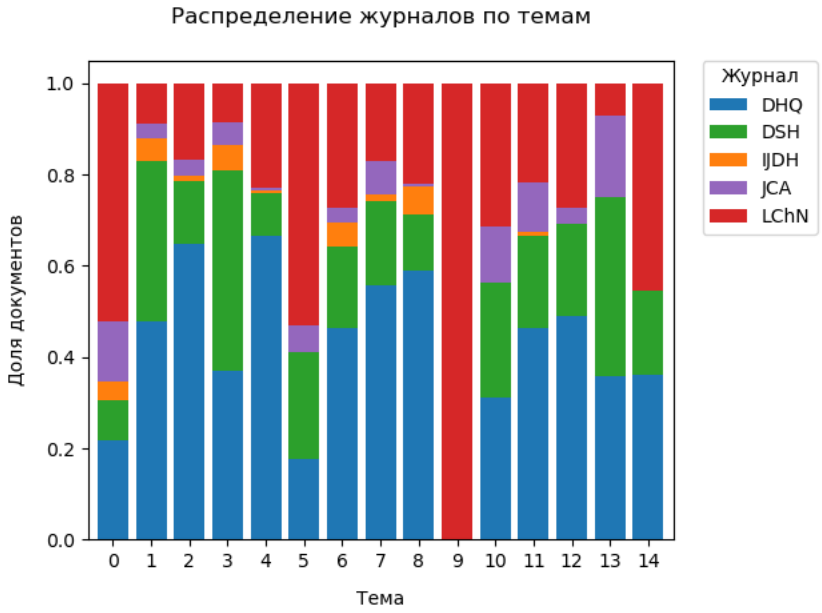

Рисунок 7 – Представленность тем по журналам

Можно заметить, что большинство выявляемых тем присутствует в каждом журнале, пусть и в разных пропорциях, что отчасти связано с неравным количеством статей между журналами. Это говорит о том, что выбранные для анализа журналы действительно объединяются общими тематиками не только формально, но и по действительному содержанию. Однако, можно заметить, что тема на рисунке под номером 9 (примерно соответствует теме 9 из исследования топ-слов для объединённых корпусов) – связанная с цифровыми исследованиями классической английской литературы, присутствует только в журнале “LChN”. Можно увидеть наиболее популярные темы для каждого журнала: так, для журнала “DSH” наиболее популярная тема 3 – “corpus, feature, character, dataset, test, category, classification, score, type, cluster, high, large” – посвящённая чисенному анализу текстовых коллекций документов, и содержащую такие ключевые для данной области слова как *кластер, корпус, набор данных, значение [метрик]*. В журнале “DHQ” самой частотной является тема 4, посвящённая осмыслению используемых в DH технологий, форматов обучения в университете и институциях (соответствует теме 11 из исследования топ-слов для объединённых корпусов).

Кроме того, можно вывести наиболее вероятные темы по каждому конкретному журналу. На рисунке 8 приведён пример для журнала “DHQ”.

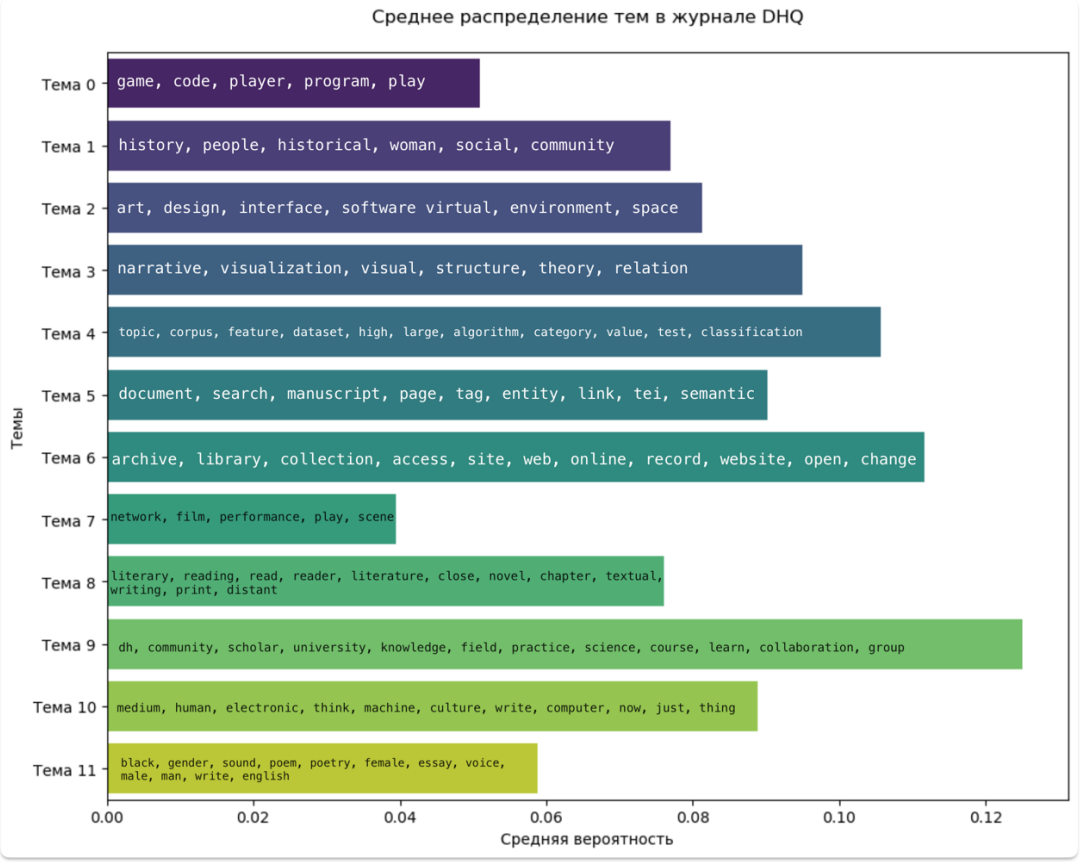

Рисунок 8 – Концентрированность темы в одном из журналов

На рисунке видно, что наиболее вероятными темами являются темы *№9* (статьи, посвященные самой научной области), *№6* (тема, посвященная архивным исследованиям) и *№4* (посвященная комьютерным средствам анализа текстов). Данные темы в немного измененном составе слов выделялись и при анализе объединённого корпуса журналов. Это говорит о широкой представленности журнала *DSH* среди всех журналов и его влиянии на структуру объединённой коллекции текстов.

Для того, чтобы посмотреть на то, какие документы корпуса близки между собой и увидеть глобальную картину подходит визуализация *UMAP* [45]. Она строится на основе матрицы вероятностей тем-документов Θ, на основе который вычисляется векторная близость на основе косинусного расстояния и происходит проекция в двумерную плоскость.

На рисунке 9 представлена визуализация близости документов по темам. Визуализация реализована интерактивной, так, что при наведении на точку выводится краткий сниппет статьи (первые *x* слов), позволяющий быстро оценить её содержание и *id* статьи для поиска её полной версии, при необходимости.

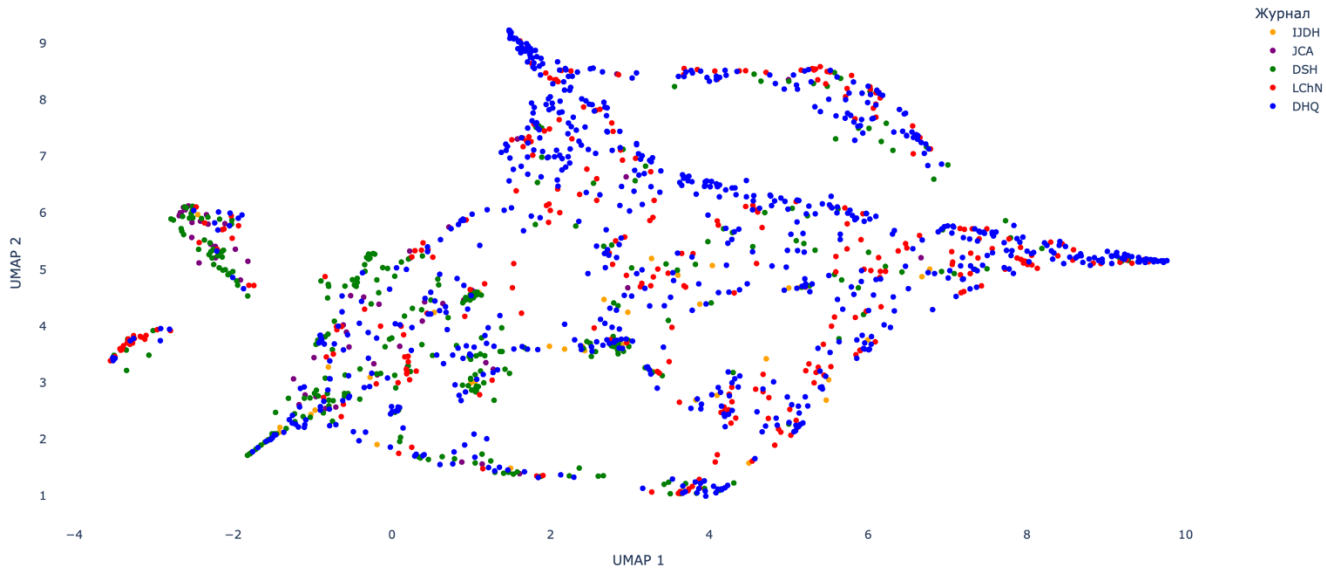

Рисунок 9 – Визуализация близости тем с помощью UMAP

Так, в правой части визуализации сосредоточены статьи, посвящённые метарефлексии над самой дисциплиной *Digital Humanities* – теме, которая, как было показано выше, преобладает в журнале *DHQ* (обозначен синим цветом). Тексты в отдельном скоплении в нижней левой части графа посвящены преимущественно исследованиям с историческим контекстом. Таким образом, визуализация позволяет выявить тематически однородные группы текстов и подтвердить ранее сделанные выводы о тематических профилях журналов.

4.2 ИНТЕРПРЕТАЦИЯ РЕЗУЛЬТАТОВ РАСШИРЕННЫХ ТЕМАТИЧЕСКИХ МОДЕЛЕЙ

4.2.1 Моделирование эволюции тем во времени

Анализ результатов динамического тематического моделирования (*DTM*), реализованного с помощью библиотеки *tomotopy*, выявил ряд особенностей, связанных с интерпретируемостью и устойчивостью извлекаемых тем. В исходных экспериментах наблюдалась тенденция к избыточной широте и несвязности тем: при переходе от одного временного среза к другому происходила почти полная смена ключевых слов, что существенно затрудняло когнитивную реконструкцию тематического ядра. В целях снижения этого эффекта были установлены пониженные значения гиперпараметров *alpha_var = 0,01, eta_var = 0,001, phi_var = 0,1*, обеспечивающих более плавные переходы между скрытыми темами в различных временных слоях.

На рисунке 10 представлены результаты анализа одной из тем, стабилизировавшейся в процессе DTM-моделирования. Как видно, значительная часть слов сохраняется в каждом из временных интервалов. Слова *digital*, *student*, *value*, *linguistic*, *collection*, *original*, *share*, *space*, *structure*, *place*, *discourse* последовательно присутствуют в большинстве временных отрезков, что указывает на высокую степень тематической устойчивости. Это позволяет говорить о наличии консистентного тематического кластера, связанного с академическим дискурсом вокруг цифровых коллекций, лингвистических данных и студенческих практик.

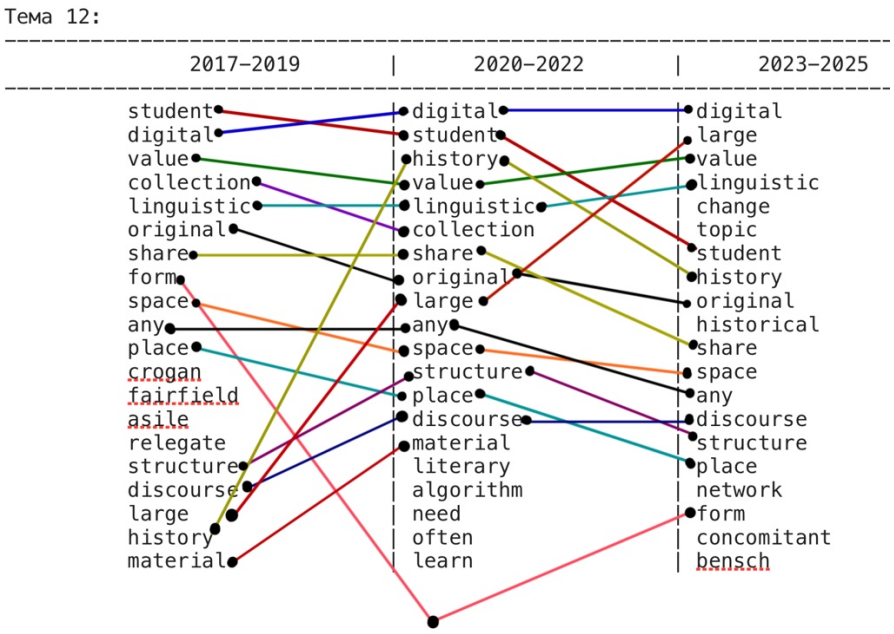

Рисунок 10 – Изменение темы по временным срезам

Наряду с устойчивостью прослеживается и динамика внутри темы. Так, в срезе 2020–2022 происходит расширение смыслового поля за счёт включения слов *algorithm*, *learn*, *что может свидетельствовать о росте интереса к алгоритмическим подходам и обучающим системам, в том числе в контексте машинного обучения*. В последующем временном интервале (2023–2025) вектор темы смещается к системным изменениям и рефлексии над структурами: в тематическое поле входят слова *change*, *topic*, *network*, *form*.

Интерес представляют слова, встречающиеся лишь единожды в одном из временных срезов. Так, *crogan, fairfield, asile* появляются только в 2017–2019, тогда как bensch и concomitant – исключительно в 2023–2025. Например, crogan и bensch отсылают к фамилиям исследователей, упоминаемых лишь в одном тексте каждый: К. Кроган – в контексте реконструкции исторических событий в компьютерных играх (два упоминания, все в статье [46]), а на Бенша ссылаются как на теоретическую опору в статье (четыре упоминания, все в статье [47]). Слово asile (фр. «убежище», восемь вхождений в одном тексте) встречается в контексте извлечения сравнений из французских литературных текстов [48], где входит в примеры обрабатываемых фрагментов. Подобные слова-маркеры

отражают локальные исследовательские фокусы и конкретные упоминания, связанные с отдельными статьями, не образующие устойчивого ядра.

Таким образом, временной анализ тем позволяет не только фиксировать устойчивое тематическое ядро, но и выявлять редкие, сопутствующие лексемы, отражающие локальные сдвиги в академическом интересе. Это делает DTM ценным инструментом как для макроанализа эволюции дисциплины, так и для точечной интерпретации смысловых новаций.

Таким образом, при корректной настройке гиперпараметров динамическое тематическое моделирование способно обеспечивать как устойчивость лексического состава тем, так и фиксацию смысловых сдвигов, расширений и контекстуальных особенностей.

4.2.2 Иерархическое тематическое моделирование

Было проведено тематическое моделирование с двухуровневой глубиной на корпусе статей журнала *Digital Scholarship in the Humanities (DSH)* с использованием иерархической модели. В данной конфигурации модели предполагается наличие нулевого уровня (гипертемы, или супертемы), первого уровня (подтемы темы) организованных в виде древовидной структуры. Число тем не задаётся и определяется самой моделью.

На начальном этапе модель демонстрировала тенденцию к объединению большинства подтем в рамках одной гипертемы, тогда как остальные гипертемы содержали лишь одну–две подтемы. Частично эта проблема смягчалась путём увеличения значений гиперпараметров α и β, однако эффект «пустых» и «переполненных» гипертем сохранялся. Более существенное улучшение удалось получить за счёт настройки специфического для данной модели параметра γ (отвечающего за *concentration coefficient* в процессе Дирихле). Увеличение значения γ с 0.1 до 1 позволило добиться более равномерного распределения подтем по супертемам. Это согласуется с

теоретическим смыслом параметра γ: его увеличение способствует росту разнообразия выделяемых тем.

Необходимость такой конфигурации модели может указывать на высокую тематическую разноплановость корпуса, наличие большого числа локальных направлений исследования и отсутствие чётко выраженной иерархической структуры. Тем не менее, в ряде случаев модель успешно выделяет логически связанные подтемы в рамках единой гипертемы.

На рисунке 11 представлен фрагмент такой иерархии для одной из супертем, к которой модель отнесла девять подтем.

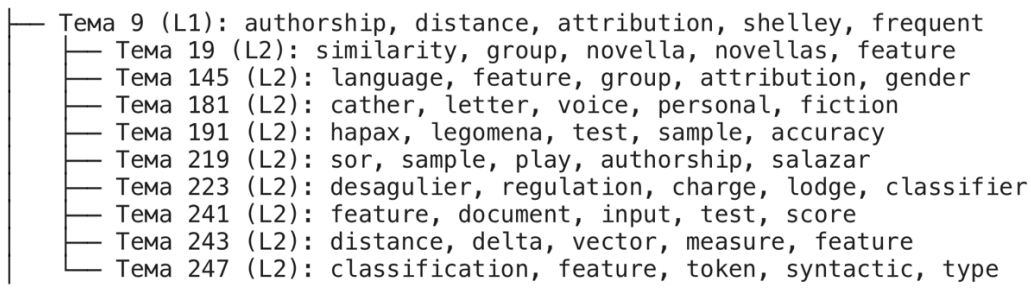

Рисунок 11 – Одна из ветвей иерархии тем, выделенных hLDA

Супертема (тема 9) фокусируется на вопросах авторства, стилометрии и тематической дистанции, а входящие в неё подтемы раскрывают отдельные аспекты данной области. При этом супертема содержит токен *shelley*, отсылающий к конкретному исследованию по определению авторства текстов, связанных с Мэри Шелли и Перси Биши Шелли [49]. Можно было бы ожидать, что данный токен окажется в одной из подтем, однако подобное распределение, вероятно, объясняется высокой частотностью слова *shelley* в корпусе, в том числе и вне текстов по стилометрии.

Анализ подтем позволяет более точно очертить исследовательские направления внутри широкой гипертемы. Так, в теме 191 встречаются *hapax* и *legomena*, формирующие терминологическую пару, обозначающую единичные случаи появления слов в корпусе, использующуюся в одной из связанных с подтемой статье [50]. Тема 19 сосредоточена на группировке текстов по признаку стилевой близости – вероятно, с акцентом на короткие литературные

формы (например, новеллы), что может использоваться при задачах авторства. Тема 181 описывает применение стилометрических методов к анализу писем [51]. Тема 219 представляет собой попытку идентификации авторства анонимной пьесы, заказанной Агустину де Салазару (отсюда – *salazar*) и предположительно адресованной сестре (исп. «*sor»*) Хуане, а тема 223 связана с фамилией компьютерного лингвиста Г. Дезагюлье. Тема 243 фокусируется на метрике *Дельта Берроуза* — одном из базовых инструментов стилометрии. В тему 241 вошли термины, составляющие техническую основу исследований, а тема 247 агрегирует методологическую базу направления.

Важно отметить, что выделяемые подтемы, в отличие от более укрупнённых тем классической *LDA*, как правило, соотносятся с одной статьёй и связаны с конкретной темой или персоналией. Несмотря на это, все подтемы остаются связанными общей темой стилометрии.

Таким образом, иерархическое тематическое моделирование позволяет при достаточной настройке параметров не только группировать тематически близкие исследования в рамках общей области, но и выявлять тематические ответвления, характерные для узких академических дискуссий.

4.2.3 Тематическое моделирование с учётом биграмм

Для данного этапа исследования были собраны корпуса статей из журналов *IJDH* и *JCA*. При подготовке текстов к тематическому моделированию все биграммы с частотой появления более пяти были объединены в единые токены формата *“слово1_слово2”*. Это преобразование позволило усилить контекстуальные связи между словами и потенциально улучшить интерпретируемость тем.

Список биграмм для объединения был сформирован из 1000 самых частотных словосочетаний по корпусу. Были применены попытки учёта биграмм, выделяемых *TF-IDF,* но специфика метода заключается в выделении слов, значимых для конкретного документа из-за их низкой встречаемости за

его пределами, а потому, выделяемые таким образом биграммы, при объединении в корпусе, не оказывали влияния на результаты тематической модели.

Результаты тематического моделирования для корпуса, подготовленного таким образом, представлены на рисунке 12.

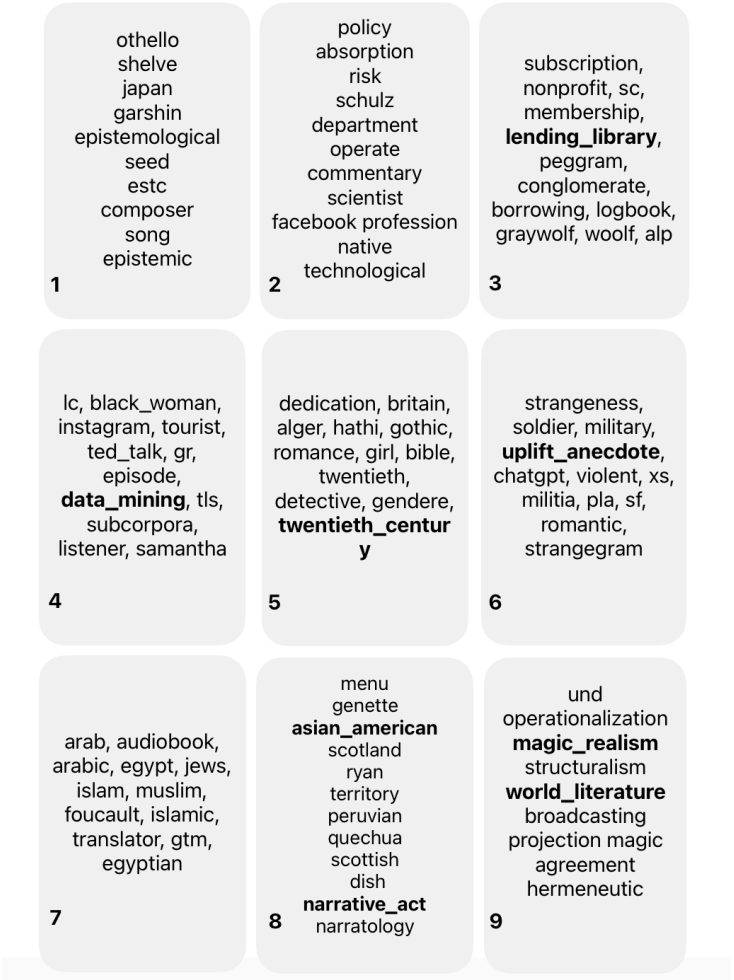

Рисунок 12 – Выделенные топ-слова в темах на корпусе с биграммами

Среди топ-слов тем удалось выделить *10* устойчивых биграмм. Их наличие способствует более точному восприятию тематической структуры текста. Так, например, биграмма *ted_talk* ясно указывает на телепередачу, а *lending_library* – на библиотечную практику. Однако наибольшее значение объединения биграмм заключается не только в повышении интерпретируемости отдельных слов, но и в перераспределении частот в

корпусе. Объединение высокочастотных словосочетаний в единые токены снижает вероятность появления их составляющих по отдельности, что позволяет «вывести на передний план» менее доминирующие, но значимые темы.

Так, среди тем были выявлены: арабо-иудейская проблематика (*тема 7*), вопросы, связанные с технологиями в контексте профессиональных и институциональных сред (*тема 2*), а также отдельные философские и литературные направления (*тема 9*). В ряде случаев состав тем оказывается трудно интерпретируемым – как, например, в *теме 6*, состоящей из таких слов, как *violence, soldier, military, chatgpt u uplift[ing] anecdote*. Проведённый ручной анализ не позволил выявить устойчивую содержательную связь между этими терминами: они преобладают в разных статьях, не пересекающихся между собой. Подобное тематическое искажение, вероятно, связано с изменением морфологии корпуса, вызванным объединением слов в биграммы.

Таким образом, проведённое исследование показало, что выявление и объединение устойчивых биграмм способствует открытию новых, ранее неочевидных тем, улучшая интерпретируемость результатов моделирования. Вместе с тем, метод требует осторожности: агрегация слов значительно трансформирует корпус и может приводить к появлению плохо интерпретируемых или несоответствующих действительности тематических связей.

4.3 ВЫВОДЫ

Для интерпретации тем использовался анализ наиболее вероятных для каждой темы текстов. Выявленные темы очерчивают содержательное поле *Digital Humanities*, охватывая как разнообразные гуманитарные дисциплины, так и технологические подходы: от работы с цифровыми архивами, базами данных и манускриптами — до исследований игр, аудиовизуальных медиа,

стилометрии, текстовых интерфейсов и культурных изменений, изучаемых количественными методами.

Кроме того, была выявлена группа публикаций, посвящённых внутренней рефлексии *DH* как научной области — включая анализ институциональной инфраструктуры, исследовательских центров, университетских программ и библиотек.

Полученные результаты подтверждают успешность применения различных подходов тематического моделирования для очерчивания предметной области *Digital Humanities* посредством анализа научных журналов.

ЗАКЛЮЧЕНИЕ

На подготовительном этапе исследования был сформирован корпус научных статей из пяти ведущих журналов по цифровым гуманитарным наукам и проведена его предобработка с учётом жанровых и лингвистических особенностей академических текстов. В теоретическом разделе была рассмотрена модель *LDA* и её расширения – динамическая (*DTM*) и иерархическая (*hLDA*) модели, а также модель *pLSA* и её развитие в методе аддитивной регуляризации тематических моделей (*ARTM*). Были разработаны программные модули для обучения моделей, сохранения, обработки и визуализации матриц распределений слов и тем, а также реализованы вычисления метрик перплексии, когерентности, и тематического разнообразия.

Проведено сравнение двух основных программных реализаций *LDA* – *Gensim* и *Tomotopy*, на основе которого был обоснован выбор *Tomotopy* как основной библиотеки. Далее была выполнена сравнительная оценка *LDA* с моделью *ARTM*. Наконец, были реализованы расширения *LDA*: *DTM* и *hLDA*, что позволило учитывать временную структуру и иерархию тем в корпусе.

Проведён ряд экспериментов по подбору оптимальных параметров моделей: гиперпараметров, числа тем и вариантов предобработки. Выявлены зависимости между метриками и параметрами моделей, что позволило обоснованно выбрать настройки, обеспечивающие наилучшую интерпретируемость результатов тематическое модели.

Наконец, были интерпретированы темы, полученные в результате моделирования, как для объединённого корпуса, так и для отдельных журналов, что позволило выявить особенности тематической структуры области цифровых гуманитарных наук.

Проведённое исследование подтвердило возможность эффективного применения тематического моделирования для анализа академических текстов.

СПИСОК ЛИТЕРАТУРЫ

1. Шерстинова Т. Ю., Кирина М. А., Москвина А.Д. Тематическое моделирование художественной прозы: оценка и интерпретируемость результатов (на примере русского рассказа 1900–1930 гг.) // Вестн. Том. гос. ун-та. Филология. 2024. №89. URL:
2. Ирхин И. А., Булатов В. Г., Воронцов К. В. Аддитивная регуляризация тематических моделей с быстрой векторизацией текста //Компьютерные исследования и моделирование. – 2020. – Т. 12. – №. 6. – С. 1515-1528.
3. Hofmann T. Probabilistic latent semantic indexing //Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. – 1999. – С. 50-57
4. Blei D. M., Ng A. Y., Jordan M. I. Latent dirichlet allocation //Journal of machine Learning research. – 2003. – Т. 3. – №. Jan. – С. 993-1022
5. Воронцов К. В. Аддитивная регуляризация тематических моделей коллекций текстовых документов //Доклады академии наук. – 2014. – Т. 456. – №. 3. – С. 268-271
6. Blei D. M., Lafferty J. D. Dynamic topic models //Proceedings of the 23rd international conference on Machine learning. – 2006. – С. 113-120.
7. Griffiths T. et al. Hierarchical topic models and the nested Chinese restaurant process //Advances in neural information processing systems. – 2003. – Т. 16. – С. 3-7.
8. Mimno D. et al. Optimizing semantic coherence in topic models //Proceedings of the 2011 conference on empirical methods in natural language processing. – 2011. – С. 262-272.
9. Newman D. et al. Automatic evaluation of topic coherence //Human language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics. – 2010. – С. 100-108.

10. Aletras N., Stevenson M. Evaluating topic coherence using distributional semantics //Proceedings of the 10th international conference on computational semantics (IWCS 2013)–Long Papers. – 2013. – С. 13-22.
11. Röder M., Both A., Hinneburg A. Exploring the space of topic coherence measures //Proceedings of the eighth ACM international conference on Web search and data mining. – 2015. – С. 399-408.
12. Webber W., Moffat A., Zobel J. A similarity measure for indefinite rankings //ACM Transactions on Information Systems (TOIS). – 2010. – Т. 28. – №. 4. – С. 1-38.
13. Sievert C., Shirley K. LDAvis: A method for visualizing and interpreting topics //Proceedings of the workshop on interactive language learning, visualization, and interfaces. – 2014. – С. 63-70.
14. Huet H., Alteri S., Taylor L. N. Manifesto: A Life on the Hyphen: Balancing Identities as Librarians, Scholars, and Digital Practitioners //DHQ: Digital Humanities Quarterly. – 2019. – Т. 13. – №. 2.
15. Eve M. P. Lessons from the library: Extreme minimalist scaling at pirate ebook platforms //Digital Humanities Quarterly. – 2022. – Т. 16. – №. 3.
16. Portela M. Multimodal editing and archival performance: a diagrammatic essay on transcoding experimental literature //Digital Humanities Quarterly. – 2014. – Т. 8. – №. 1.
17. Schoenbeck R. Playing with Chance: On Random Generation in Playable Media and Electronic Literature //DHQ: Digital Humanities Quarterly. – 2013. – Т. 7. – №. 3.
18. Eskelinen M. Cybertext poetics: the critical landscape of new media literary theory. – Bloomsbury Publishing USA, 2012.
19. Seminck O. et al. The evolution of the idiolect over the lifetime: A quantitative and qualitative study of french 19th century literature //Journal of Cultural Analytics. – 2022. – Т. 7. – №. 3.

20. deTombe J. Digital editing as autopoietic process //Digital Studies/Le champ numérique. – 2016. – Т. 6. – №. 6.
21. Broyles P. A. Digital Editions and Version Numbering //DHQ: Digital Humanities Quarterly. – 2020. – Т. 14. – №. 2.
22. Have I., Enevoldsen K. From close listening to distant listening: Developing tools for Speech-Music discrimination of Danish music radio //DHQ: Digital Humanities Quarterly. – 2021. – №. 1.
23. Malmstedt J. Rhythms of silence: digital audio analysis of Swedish radio broadcasting, 1980-1989 //Journal of Cultural Analytics. – 2022. – Т. 7. – №. 1. – С. 108-138.
24. Woolhouse M., Renwick J. Generalizing case-based analyses in the study of global music consumption //Digital Studies/Le champ numérique. – 2016. – Т. 5. – №. 3.
25. Mittell J. Deformin'in the Rain: How (and Why) to Break a Classic Film //DHQ: Digital Humanities Quarterly. – 2021. – Т. 15. – №. 1.
26. Siemens R. Subsidium: Master List of REKn Primary Sources //Digital Studies/Le champ numérique. – 2011. – Т. 2. – №. 2.
27. Freebury-Jones D. “Fearful Dreams” in Thomas Kyd’s Restored Canon //Digital Studies/Le champ numérique. – 2019. – Т. 9. – №. 1.
28. Wójcik J. Cluster Analysis in Tracing Textual Dependencies–A Case of Psalm 6 in 16th-century English Devotional Manuals //Digital Humanities Quarterly. – 2023. – Т. 17. – С. 1-16.
29. Estill L., Meneses L. Is Falstaff Falstaff? Is Prince Hal Henry V?: Topic Modeling Shakespeare’s Plays //Digital Studies/Le champ numérique. – 2018. – Т. 8. – №. 1.
30. Cordell R. "Taken Possession of": The Reprinting and Reauthorship of Hawthorne's" Celestial Railroad" in the Antebellum Religious Press //DHQ: Digital Humanities Quarterly. – 2013. – Т. 7. – №. 1.

31. Hołobut A., Rybicki J. The stylometry of film dialogue: Pros and pitfalls. – 2020.
32. Evans M., Hogarth A. Stylistic palimpsests: Computational stylistic perspectives on precursory authorship in Aphra Behn’s drama //Digital Scholarship in the Humanities. – 2021. – Т. 36. – №. 1. – С. 64-86.
33. Evert S. et al. Understanding and explaining Delta measures for authorship attribution //Digital Scholarship in the Humanities. – 2017. – Т. 32. – №. suppl_2. – С. ii4-ii16.
34. Craig H. Principal components analysis in stylometry //Digital Scholarship in the Humanities. – 2024. – Т. 39. – №. 1. – С. 97-108.
35. Hoover D. L. The microanalysis of style variation //Digital Scholarship in the Humanities. – 2017. – Т. 32. – №. suppl_2. – С. ii17-ii30.
36. Christian-Lamb C., Shrout A. H. " Starting From Scratch"? Workshopping New Directions in Undergraduate Digital Humanities //DHQ: Digital Humanities Quarterly. – 2017. – Т. 11. – №. 3.
37. Kasten-Mutkus K., Costello L., Chase D. Raising Visibility in the Digital Humanities Landscape: Academic Engagement and the Question of the Library's Role //DHQ: Digital Humanities Quarterly. – 2019. – Т. 13. – №. 2.
38. El Khatib R. et al. An “Open Lab?” The Electronic Textual Cultures Lab in the Evolving Digital Humanities Landscape. – 2020.
39. Siemens L. Starting and Sustaining Digital Humanities/Digital Scholarships Centers: Lessons from the Trenches //Digit. Humanit. Q. – 2023. – Т. 17. – №. 3.
40. Almas B., Clérice T. Continuous integration and unit testing of digital editions //Digital Humanities Quarterly. – 2018. – Т. 11. – №. 4.
41. Kizhner I. et al. Digital Humanities at Siberian Federal University //Digital Studies/Le champ numérique. – 2015. – Т. 6. – №. 2.

42. Berensmeyer I., Trurnit S. Post-War British Women Writers and their Cultural Impact: A Quantitative Approach //Journal of Cultural Analytics. – 2022. – T. 7. – №. 1. – C. 81-107.
43. Antoniak M. et al. The afterlives of shakespeare and company in online social readership //arXiv preprint arXiv:2401.07340. – 2024.
44. Karmanov F., Kotin J. A Counterfactual Canon //Journal of Cultural Analytics. – 2024. – T. 9. – №. 2.
45. McInnes L., Healy J., Melville J. Umap: Uniform manifold approximation and projection for dimension reduction //arXiv preprint arXiv:1802.03426. – 2018. – C. 17-51.
46. Schreibman S., Papadopoulos C. Textuality in 3D: Three-dimensional (re) constructions as digital scholarly editions //International Journal of Digital Humanities. – 2019. – T. 1. – №. 2. – C. 221-233.
47. Hekanaho L., Hirvonen M., Virtanen T. Language-based machine perception: linguistic perspectives on the compilation of captioning datasets //Digital Scholarship in the Humanities. – 2024. – T. 39. – №. 3. – C. 864-883.
48. At the crossroads between the scientific and the literary discourse
49. Suddaby L., Ross G. J. Did Mary Shelley write Frankenstein? A stylometric analysis //Digital Scholarship in the Humanities. – 2023. – T. 38. – №. 2. – C. 750-765.
50. Faltýnek D., Matlach V. Hapax remains: Regularity of low-frequency words in authorial texts //Digital Scholarship in the Humanities. – 2022. – T. 37. – №. 3. – C. 693-715.
51. Exploring the Intersection of Personal and Public Authorial Voice in the Works of Willa Cather

ПРИЛОЖЕНИЕ

Основной модуль тематического моделирования

```
from pathlib import Path
import json
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import umap.umap_ as umap
import plotly.express as px
from plotly.offline import plot
import webbrowser
from collections import defaultdict, Counter
from itertools import combinations
from math import log
import pyLDAvis
import pyLDAvis.gensim_models as gensimvis
from sklearn.feature_extraction.text import CountVectorizer
from gensim.corpora import Dictionary
from gensim.models import LdaModel
import tomotopy
from tomotopy.coherence import Coherence

# Константы
NUM_TOPICS = 15
TOP_K = 10
TOP_DOCS = 5
PASSES = 1
WINDOW_SIZE = 20
CV_WINDOW_SIZE = 140
ITERATIONS = 100

ALPHA = 1
BETA = 0.001
MIN_CF = 1
MIN_DF = 1
RM_TOP = 10

RBD_P = 0.9
EPS_PHI = 1e-6
EPS_THETA = 1e-5
EPS = 1e-8
ROUND = 4
RANDOM_STATE = 20

CORPUS_NAME = "Test"
INPUT_JSON_PATH = Path("preprocessed") / f"{CORPUS_NAME}.json"
TM_INPUT_ROOT = Path("tm_input")
TM_OUTPUT_ROOT = Path("tm_output")

JOURNAL_COLORS = {
    'DHQ': 'blue',
    'DSH': 'green',
    'LChN': 'red',
    'JCA': 'purple',
    'IJDH': 'orange'
}

def load_and_preprocess_data(input_path):
    """загрузка и предварительная обработка данных"""
    with open(input_path, encoding="utf-8") as f:
        raw_docs = json.load(f)

    docs_meta = {doc["id"]: doc for doc in raw_docs if doc.get("text")}
    tokenized_docs = [doc["text"].split() for doc in raw_docs if doc.get("text")]
```

```
    return tokenized_docs, docs_meta

def prepare_gensim_input(tokenized_docs, output_dir):
    """подготовка входных данных для gensim"""
    output_dir.mkdir(parents=True, exist_ok=True)
    dictionary = Dictionary(tokenized_docs)
    bow_corpus = [dictionary.doc2bow(doc) for doc in tokenized_docs]

    dictionary.save(str(output_dir / "model.dict"))
    with open(output_dir / "bow_corpus.json", "w", encoding="utf-8") as f:
        json.dump([[list(pair) for pair in doc] for doc in bow_corpus], f,
ensure_ascii=False)

    return dictionary, bow_corpus

def prepare_tomotopy_input(tokenized_docs, output_dir):
    """подготовка входных данных для tomotopy"""
    output_dir.mkdir(parents=True, exist_ok=True)
    with open(output_dir / "tokenized_docs.json", "w", encoding="utf-8") as f:
        json.dump(tokenized_docs, f, ensure_ascii=False, indent=2)

    return tokenized_docs

def run_gensim_model(tokenized_docs, output_dir, num_topics=NUM_TOPICS,
iterations=ITERATIONS,
                     passes=PASSES, alpha='symmetric', eta=None, top_k=TOP_K, no_below=1,
                     no_above=1.0, random_state=RANDOM_STATE):

    dictionary = Dictionary(tokenized_docs)
    dictionary.filter_extremes(no_below=no_below, no_above=no_above)
    bow_corpus = [dictionary.doc2bow(doc) for doc in tokenized_docs]

    lda_model = LdaModel(
        corpus=bow_corpus,
        id2word=dictionary,
        num_topics=num_topics,
        passes=1,
        iterations=iterations,
        alpha=alpha,
        eta=eta,
        random_state=random_state
    )

    for pass_num in range(1, passes + 1):
        lda_model.update(bow_corpus)
        print_iteration_topics(lda_model, pass_num, "gensim")

    theta = []
    for doc_bow in bow_corpus:
        dist = lda_model.get_document_topics(doc_bow, minimum_probability=0.0)
        theta.append([float(prob) for _, prob in sorted(dist)])

    phi = lda_model.get_topics()

    return lda_model, np.array(theta), np.array(phi), dictionary, bow_corpus

def run_tomotopy_model(tokenized_docs, output_dir, model_type='lda',
num_topics=NUM_TOPICS,
                       iterations=ITERATIONS, top_k=TOP_K, alpha=ALPHA, eta=BETA,
min_cf=MIN_CF,
                       min_df=MIN_DF, rm_top=RM_TOP, seed=RANDOM_STATE):

    model = tomotopy.LDAModel(
        k=num_topics,
        alpha=alpha,
        eta=eta,
        seed=seed,
        min_cf=min_cf,
```

```
        min_df=min_df,
        rm_top=rm_top
    )

    for doc in tokenized_docs:
        model.add_doc(doc)

    for i in range(iterations):
        model.train(1)
        if i % 10 == 0 or i == iterations - 1:
            print_iteration_topics(model, i, "tomotopy")

    word_to_id = {word: i for i, word in enumerate(model.used_vocabs)}

    phi = []
    for k in range(model.k):
        topic_words = model.get_topic_words(k, top_n=len(model.used_vocabs))
        topic_dist = [0.0] * len(model.used_vocabs)
        for word, prob in topic_words:
            topic_dist[word_to_id[word]] = float(prob)
        phi.append(topic_dist)

    bow_corpus = []
    for doc in model.docs:
        doc_bow = defaultdict(int)
        for word in doc.words:
            doc_bow[word_to_id[model.used_vocabs[word]]] += 1
        bow_corpus.append(list(doc_bow.items()))

    theta = [doc.get_topic_dist() for doc in model.docs]

    return model, np.array(theta), np.array(phi), word_to_id, bow_corpus

def compute_sparsity(matrix, epsilon):
    """вычисление разреженности матрицы"""
    sparsity = float(np.mean(matrix < epsilon))
    return round(sparsity, ROUND)

def compute_entropy_phi(phi):
    """энтропия для матрицы phi"""
    eps = EPS
    entropy = -np.sum(phi * np.log(phi + eps), axis=1)
    return round(np.mean(entropy), ROUND)

def compute_entropy_theta(theta):
    """энтропия для матрицы theta"""
    eps = EPS
    entropy = -np.sum(theta * np.log(theta + eps), axis=1)
    return round(np.mean(entropy), ROUND)

def compute_topic_diversity(phi, top_k=TOP_K):
    """вычисление разнообразия тем"""
    top_words = [set(np.argsort(topic)[::-1][:top_k]) for topic in phi]
    unique_words = set.union(*top_words)
    diversity = len(unique_words) / (top_k * len(phi))
    return round(diversity, 4)

def compute_rbd(phi, top_k=TOP_K, p=RBD_P):
    """вычисление rbd"""
    topic_rankings = [np.argsort(-topic)[:top_k].tolist() for topic in phi]
    rbo_scores = []

    for a, b in combinations(topic_rankings, 2):
        score = 0.0
        for d in range(1, len(a) + 1):
            agreement = len(set(a[:d]) & set(b[:d]))
            score += agreement / d * pow(p, d)
        rbo = (1 - p) * score
```

```
            rbo_scores.append(1 - rbo)

        rbd = float(np.mean(rbo_scores)) if rbo_scores else 0.0
        return round(rbd, ROUND)

    def compute_norm_rbd(phi, top_k=TOP_K, p=RBD_P):
        """вычисление нормализованного rbd"""
        topic_rankings = [np.argsort(-topic)[:top_k].tolist() for topic in phi]
        rbd_scores = []

        max_score = sum(p**d for d in range(1, top_k + 1))

        for a, b in combinations(topic_rankings, 2):
            score = 0.0
            for d in range(1, top_k + 1):
                agreement = len(set(a[:d]) & set(b[:d]))
                score += agreement / d * p**d

            rbo = score / max_score if max_score > 0 else 0.0
            rbd_scores.append(1 - rbo)

        return round(float(np.mean(rbd_scores)) if rbd_scores else 0.0, ROUND)

    def compute_perplexity(model, theta, phi, bow_corpus):
        """расчет перплексии"""
        log_likelihood = 0
        total_words = 0

        for doc_idx, doc in enumerate(bow_corpus):
            doc_words = 0
            doc_log_prob = 0

            for word_id, count in doc:
                word_prob = 0
                for topic_idx in range(theta.shape[1]):
                    word_prob += theta[doc_idx, topic_idx] * phi[topic_idx, word_id]

                if word_prob > 0:
                    doc_log_prob += count * np.log(word_prob)
                doc_words += count

            log_likelihood += doc_log_prob
            total_words += doc_words

        if total_words == 0:
            return float('nan')

        perplexity = np.exp(-log_likelihood / total_words)
        return round(perplexity, ROUND)

    def compute_coherence_metrics(phi, bow_corpus, vocab=None, top_k=TOP_K,
    tokenized_docs=None):
        """вычисление метрик когерентности"""
        inv_vocab = {v: k for k, v in vocab.items()} if
    isinstance(next(iter(vocab.values())), int) else vocab
        eps = 1e-8

        def compute_umass(phi, bow_corpus, top_k=TOP_K, eps=EPS):
            D = len(bow_corpus)
            scores = []
            for topic in phi:
                top_words = np.argsort(-topic)[:top_k]
                score = 0.0
                pairs = 0
                for i in range(1, top_k):
                    for j in range(i):
                        wi, wj = top_words[i], top_words[j]
                        doc_wj = sum(1 for doc in bow_corpus if any(w == wj for w, _ in doc))
```

```
                    doc_wi_wj = sum(1 for doc in bow_corpus if all(w in [w_ for w_, _ in
doc] for w in [wi, wj]))
                    if doc_wj > 0:
                        score += log((doc_wi_wj + eps) / doc_wj)
                        pairs += 1
            scores.append(score / pairs if pairs else 0)
        return round(float(np.mean(scores)), 4) if scores else 0.0

    def compute_uci(phi, texts, top_k=TOP_K, window_size=WINDOW_SIZE, eps=EPS):
        word_counts = defaultdict(int) # сколько раз слово появилось в каком-либо окне
        pair_counts = defaultdict(int) # сколько раз пара слов появилась вместе в каком-
либо окне
        total_windows = 0 # общее количество окон

        for doc in texts: # по каждому документу все возможные окна длины window_size
            for i in range(len(doc) - window_size + 1):
                window = doc[i:i + window_size]
                unique_words = list(set(window)) # уникальные слова, без повторов(set)
                for w in unique_words: # подсчет частот слов
                    word_counts[w] += 1
                for i_w in range(len(unique_words)): # подсчет частот пар слов
                    for j_w in range(i_w):
                        pair = (unique_words[i_w], unique_words[j_w])
                        pair_counts[pair] += 1
                total_windows += 1

        scores = []
        for topic in phi:
            top_words = np.argsort(-topic)[:top_k]
            topic_score = 0.0
            pairs = 0
            for i in range(1, top_k): # подсчет вероятностей для формулы когерентности
                for j in range(i):
                    w_i, w_j = top_words[i], top_words[j]
                    pair = (w_i, w_j) if w_i < w_j else (w_j, w_i)
                    p_i = word_counts.get(w_i, 0) / total_windows
                    p_j = word_counts.get(w_j, 0) / total_windows
                    p_ij = pair_counts.get(pair, 0) / total_windows
                    if p_i > 0 and p_j > 0:
                        topic_score += np.log((p_ij + eps) / (p_i * p_j))
                        pairs += 1
            scores.append(topic_score / pairs if pairs else 0) # усреднение по парам
        return np.mean(scores) if scores else 0.0 # усреднение по числу тем

    def compute_npmi(phi, bow_corpus, top_k=TOP_K, eps=eps):
        D = len(bow_corpus)
        scores = []
        for topic in phi:
            top_words = np.argsort(-topic)[:top_k]
            score = 0.0
            pairs = 0
            for i in range(1, top_k):
                for j in range(i):
                    wi, wj = top_words[i], top_words[j]
                    doc_wi = sum(1 for doc in bow_corpus if any(w == wi for w, _ in doc))
                    doc_wj = sum(1 for doc in bow_corpus if any(w == wj for w, _ in doc))
                    doc_wi_wj = sum(1 for doc in bow_corpus if all(w in [w_ for w_, _ in
doc] for w in [wi, wj]))
                    p_i = doc_wi / D
                    p_j = doc_wj / D
                    p_ij = doc_wi_wj / D
                    if p_ij > 0:
                        npmi = log(p_ij / (p_i * p_j + eps)) / -log(p_ij + eps)
                        score += npmi
                        pairs += 1
            scores.append(score / pairs if pairs else 0)
        return round(float(np.mean(scores)), 4) if scores else 0.0
```

```
    def compute_cv_window(phi, tokenized_docs, vocab, top_k=TOP_K,
window_size=CV_WINDOW_SIZE):
        cooccur = defaultdict(Counter)
        all_words = set()

        for doc in tokenized_docs:
            for i in range(len(doc) - window_size + 1):
                window = doc[i:i+window_size]
                for w1, w2 in combinations(window, 2):
                    if w1 == w2:
                        continue
                    cooccur[w1][w2] += 1
                    cooccur[w2][w1] += 1
                    all_words.update([w1, w2])

        word_list = list(all_words)
        word2idx = {word: i for i, word in enumerate(word_list)}

        co_matrix = np.zeros((len(word_list), len(word_list)))

        for w1 in cooccur:
            for w2 in cooccur[w1]:
                i, j = word2idx[w1], word2idx[w2]
                co_matrix[i, j] = cooccur[w1][w2]

        cv_scores = []

        for topic_vec in phi:
            top_indices = np.argsort(-np.array(topic_vec))[:top_k]
            top_words = [vocab.get(idx, None) for idx in top_indices if idx in vocab]

            vectors = []
            for word in top_words:
                if word not in word2idx:
                    continue
                idx = word2idx[word]
                vectors.append(co_matrix[idx])

            sims = []
            for i in range(len(vectors)):
                for j in range(i + 1, len(vectors)):
                    v1 = vectors[i]
                    v2 = vectors[j]
                    denom = np.linalg.norm(v1) * np.linalg.norm(v2)
                    if denom == 0:
                        continue
                    sim = np.dot(v1, v2) / denom
                    sims.append(sim)

            if sims:
                avg_sim = np.mean(sims)
                cv_scores.append(avg_sim)

        return round(float(np.mean(cv_scores)) if cv_scores else 0.0, ROUND)

    umass = compute_umass(phi, bow_corpus, top_k=TOP_K)
    uci = compute_uci(phi, bow_corpus, top_k=TOP_K)
    npmi = compute_npmi(phi, bow_corpus, top_k=TOP_K)
    if tokenized_docs is not None:
        cv_win = compute_cv_window(phi, tokenized_docs, vocab, top_k=TOP_K)
    else:
        cv_win = None

    return {
        "umass": umass,
        "uci": uci,
        "npmi": npmi,
        "cv_win": cv_win
```

```
    }

def compute_all_metrics(model, theta, phi, bow_corpus=None, dictionary=None,
tokenized_docs=None):
    metrics = {
        "sparsity_theta": compute_sparsity(theta, EPS_THETA),
        "sparsity_phi": compute_sparsity(phi, EPS_PHI),
        "entropy_phi": compute_entropy_phi(phi),
        "entropy_theta": compute_entropy_theta(theta),
        "topic_diversity": compute_topic_diversity(phi),
        "rbd": compute_rbd(phi),
        "norm_rbd": compute_norm_rbd(phi)
    }

    if bow_corpus is not None:
        metrics["perplexity"] = compute_perplexity(model, theta, phi, bow_corpus)

        if dictionary is not None:
            if isinstance(dictionary, dict):
                vocab = dictionary
            else:
                vocab = {v: k for k, v in dictionary.items()}
            metrics.update(compute_coherence_metrics(phi, bow_corpus, vocab,
tokenized_docs=tokenized_docs))

    return metrics

def compute_tomotopy_metrics(model, tokenized_docs=None):
    """вычисление всех метрик tomotopy"""
    metrics = {}

    metrics['coherence'] = compute_all_coherence_metrics(model, tokenized_docs)

    metrics.update({
        'perplexity': getattr(model, 'perplexity', None),
        'log_likelihood': getattr(model, 'll_per_word', None),
        'num_topics': model.k,
        'num_words': len(model.used_vocabs)
    })

    try:
        metrics['topic_diversity'] = compute_topic_diversity(model.get_topic_word_dist())
    except Exception as e:
        print(f"Error calculating topic diversity: {e}")
        metrics['topic_diversity'] = None

    return metrics

def compute_all_coherence_metrics(model, tokenized_docs=None, top_n=TOP_K):
    """вычисление 4 метрик когерентности для tomotopy"""
    coherence_metrics = {}

    metrics_config = {
        'u_mass': {'coherence': 'u_mass', 'window_size': 0},
        'c_uci': {'coherence': 'c_uci', 'window_size': WINDOW_SIZE},
        'c_npmi': {'coherence': 'c_npmi', 'window_size': WINDOW_SIZE},
        'c_v': {'coherence': 'c_v', 'window_size': CV_WINDOW_SIZE}
    }

    for metric_name, config in metrics_config.items():
        try:
            coh = Coherence(
                corpus=model,
                coherence=config['coherence'],
                window_size=config['window_size'],
                top_n=top_n
            )
```

```
            avg_score = coh.get_score()
            topic_scores = []
            for k in range(model.k):
                topic_scores.append(coh.get_score(topic_id=k))

            coherence_metrics[metric_name] = {
                'average': avg_score,
                'per_topic': topic_scores,
                'std': np.std(topic_scores)
            }

        except Exception as e:
            print(f"Error calculating {metric_name}: {e}")
            coherence_metrics[metric_name] = None

    return coherence_metrics

def visualize_model(model, bow_corpus=None, dictionary=None, output_dir=None):
    """визуализация модели"""
    if output_dir is None:
        output_dir = Path(".")

    if hasattr(model, 'show_topics') and bow_corpus is not None and dictionary is not
None:
        vis_data = gensimvis.prepare(model, bow_corpus, dictionary)

    elif hasattr(model, 'used_vocabs'): # для tomotopy
        vocab = list(model.used_vocabs)
        vocab_index = {word: i for i, word in enumerate(vocab)}
        term_freq = defaultdict(int)

        for doc in model.docs:
            for word in doc.words:
                term_freq[word] += 1

        phi = []
        for k in range(model.k):
            topic_dist = [0.0] * len(vocab)
            for word, prob in model.get_topic_words(k, top_n=len(vocab)):
                topic_dist[vocab_index[word]] = prob
            phi.append(topic_dist)
        phi = np.array(phi)

        theta = np.array([doc.get_topic_dist() for doc in model.docs])
        doc_lengths = [len(doc.words) for doc in model.docs]
        term_frequency = [term_freq[word] for word in vocab]

        vis_data = pyLDAvis.prepare(
            topic_term_dists=phi,
            doc_topic_dists=theta,
            doc_lengths=doc_lengths,
            vocab=vocab,
            term_frequency=term_frequency
        )

    else:
        raise ValueError("Unsupported model type or missing parameters")

    vis_path = output_dir / "pyldavis.html"
    pyLDAvis.save_html(vis_data, str(vis_path))
    print(f"Visualization saved to: {vis_path}")

    return vis_data

def save_matrices(theta, phi, output_dir):
    """сохранение матриц theta и phi"""
    output_dir.mkdir(parents=True, exist_ok=True)
    with open(output_dir / "theta.json", "w", encoding="utf-8") as f:
```

```
        json.dump(theta.tolist(), f, indent=2)
    with open(output_dir / "phi.json", "w", encoding="utf-8") as f:
        json.dump(phi.tolist(), f, indent=2)

def save_metrics(metrics, output_dir):
    """сохранение метрик"""
    serializable_metrics = {k: float(v) if isinstance(v, (np.float32, np.float64)) else v
                            for k, v in metrics.items()}

    with open(output_dir / "metrics.json", "w", encoding="utf-8") as f:
        json.dump(serializable_metrics, f, indent=2)

def save_top_words(model, output_dir, top_n=TOP_K):
    """сохранение топ-слов для каждой темы"""
    top_words_path = output_dir / "top_words.txt"

    with open(top_words_path, 'w', encoding='utf-8') as f:
        if hasattr(model, 'show_topics'):
            for topic_id in range(model.num_topics):
                topic_words = model.show_topic(topic_id, topn=top_n)
                words = [word for word, prob in topic_words]
                f.write(f"Topic {topic_id}: {', '.join(words)}\n")

        elif hasattr(model, 'get_topic_words'):
            for topic_id in range(model.k):
                topic_words = model.get_topic_words(topic_id, top_n=top_n)
                words = [word for word, prob in topic_words]
                f.write(f"Topic {topic_id}: {', '.join(words)}\n")

    print(f"Top words saved to {top_words_path}")

def save_top_documents(theta, docs_meta, output_dir, top_n=TOP_DOCS):
    """сохранение топ-документов для каждой темы"""
    top_docs_path = output_dir / "top_documents.txt"
    theta_array = np.array(theta)

    with open(top_docs_path, 'w', encoding='utf-8') as f:
        for topic_id in range(theta_array.shape[1]):
            top_doc_indices = np.argsort(-theta_array[:, topic_id])[:top_n]

            f.write(f"\n=== Topic {topic_id} ===\n")

            for i, doc_idx in enumerate(top_doc_indices):
                doc_id = list(docs_meta.keys())[doc_idx] if docs_meta else str(doc_idx)
                doc_info = docs_meta.get(doc_id, {})

                text = doc_info.get('text', '')
                snippet = ' '.join(text.split()[:20]) + ('...' if len(text.split()) > 20
else '')

                f.write(
                    f"\nDocument #{i+1} (ID: {doc_id})\n"
                    f"URL: {doc_info.get('url', 'not specified')}\n"
                    f"Snippet: {snippet}\n"
                    f"Topic weight: {theta_array[doc_idx, topic_id]:.4f}\n"
                )

    print(f"Top documents saved to {top_docs_path}")

def print_iteration_topics(model, iteration, model_type):
    """вывод топ-слов на текущей итерации"""
    if model_type == "gensim":
        topics = model.show_topics(num_topics=-1, num_words=TOP_K, formatted=False)
        for topic_id, topic_words in topics:
            topic_str = " + ".join([f"{prob:.3f}*\"{word}\"" for word, prob in
topic_words])
            print(f"Topic {topic_id}: {topic_str}")
```

```
    elif model_type == "tomotopy":
        if iteration % 10 == 0 or iteration == ITERATIONS - 1:
            for topic_id in range(model.k):
                topic_words = model.get_topic_words(topic_id, top_n=TOP_K)
                topic_str = " + ".join([f"{prob:.3f}*\"{word}\"" for word, prob in
topic_words])
                print(f"Topic {topic_id}: {topic_str}")

def run_pipeline(model_type, tomotopy_model=None):
    """пайплайн тематического моделирования"""
    if model_type == "tomotopy" and not tomotopy_model:
        tomotopy_model = "lda"

    tokenized_docs, docs_meta = load_and_preprocess_data(INPUT_JSON_PATH)

    output_dir_name = f"{model_type}_{tomotopy_model if model_type == 'tomotopy' else
'lda'}_{CORPUS_NAME}_K_{NUM_TOPICS}"
    output_dir = TM_OUTPUT_ROOT / output_dir_name
    output_dir.mkdir(parents=True, exist_ok=True)

    if model_type == "gensim":
        model, theta, phi, dictionary, bow_corpus = run_gensim_model(
            tokenized_docs=tokenized_docs,
            output_dir=output_dir,
            num_topics=NUM_TOPICS,
            iterations=ITERATIONS,
            passes=PASSES,
            alpha='auto',
            eta='auto',
            top_k=TOP_K,
            no_below=1,
            no_above=1.0
        )
        visualize_model(model, bow_corpus, dictionary, output_dir=output_dir)
        metrics = compute_all_metrics(model, theta, phi, bow_corpus=bow_corpus,
                        dictionary=dictionary, tokenized_docs=tokenized_docs)

    elif model_type == "tomotopy":
        model, theta, phi, word_to_id, bow_corpus = run_tomotopy_model(
            tokenized_docs=tokenized_docs,
            output_dir=output_dir,
            model_type=tomotopy_model,
            num_topics=NUM_TOPICS,
            iterations=ITERATIONS,
            top_k=TOP_K,
            alpha=ALPHA,
            eta=BETA,
            min_cf=MIN_CF,
            min_df=MIN_DF,
            rm_top=RM_TOP
        )
        visualize_model(model, output_dir=output_dir)
        tomotopy_metrics = compute_tomotopy_metrics(model, tokenized_docs)
        save_tomotopy_metrics(tomotopy_metrics, output_dir)

        id_to_word = {v: k for k, v in word_to_id.items()}
        metrics = compute_all_metrics(
            model,
            theta,
            phi,
            bow_corpus=bow_corpus,
            dictionary=id_to_word,
            tokenized_docs=tokenized_docs
        )

    save_matrices(theta, phi, output_dir)
    save_metrics(metrics, output_dir)
    save_top_words(model, output_dir)
```

```
    save_top_documents(theta, docs_meta, output_dir)

    print(f"Результаты после успешного обучения сохранены в папку: {output_dir}")

def analyze_journals():
    """анализ журналов с помощью tomotopy lda"""
    all_docs, journal_docs = load_data(INPUT_JSON_PATH)
    docs_meta = [doc for doc in all_docs if doc.get('text')]

    print("\nAnalyzing full corpus")
    tokenized_all = preprocess_documents(journal_docs['ALL'])
    model_all = train_tomotopy_model(tokenized_all)

    print("\nTop words for full corpus:")
    print_top_words(model_all)

    theta_all = get_topic_document_distribution(model_all)
    plot_journal_distribution_by_topic(docs_meta, theta_all)

    journal_topics = {
        'DHQ': 12,
        'DSH': 10,
        'LChN': 11,
        'JCA': 7,
        'IJDH': 4
    }

    journal_models = {}
    for journal, num_topics in journal_topics.items():
        model, theta = analyze_journal(journal, journal_docs[journal], num_topics)
        journal_models[journal] = model

    visualize_umap(docs_meta, theta_all)

def analyze_journal(journal_name, journal_docs, num_topics):
    """анализ отдельного журнала"""
    print(f"\nЖурнал: {journal_name}")
    print(f"Число документов: {len(journal_docs)}")

    tokenized_docs = preprocess_documents(journal_docs)
    model = train_tomotopy_model(tokenized_docs, num_topics)

    print("\nTop words by topic:")
    print_top_words(model)

    perplexity = model.perplexity
    coherence = calculate_coherence(model, tokenized_docs)

    print("\nModel metrics:")
    print(f"Perplexity: {perplexity:.4f}")
    print(f"Coherence: {coherence:.4f}")

    theta = get_topic_document_distribution(model)

    df = pd.DataFrame({
        'topic': [f"{i}" for i in range(num_topics)],
        'mean_prob': theta.mean(axis=0)
    })

    plt.figure(figsize=(10, 8))
    ax = sns.barplot(
        data=df,
        y='topic',
        x='mean_prob',
        hue='topic',
        dodge=False,
        palette='viridis',
        orient='h'
```

```
    )

    plt.title(f'Average topic distribution in {journal_name}', pad=20)
    plt.xlabel('Average probability')
    plt.ylabel('Topics')
    plt.yticks(rotation=0)
    plt.legend().remove()
    plt.tight_layout()
    plt.show()

    return model, theta

def plot_journal_distribution_by_topic(docs_meta, theta, num_topics=NUM_TOPICS):
    """визуализация распределения журналов по темам"""
    doc_ids = [doc['id'] for doc in docs_meta]
    journal_topics = []

    for i, doc_id in enumerate(doc_ids):
        journal = docs_meta[i]['journal']
        topic_dist = theta[i]
        dominant_topic = np.argmax(topic_dist)
        journal_topics.append({'journal': journal, 'topic': dominant_topic})

    df = pd.DataFrame(journal_topics)
    topic_journal_counts = df.groupby(['topic', 'journal']).size().unstack().fillna(0)
    topic_journal_normalized = topic_journal_counts.div(topic_journal_counts.sum(axis=1),
axis=0)

    plt.figure(figsize=(12, 8))
    ax = topic_journal_normalized.plot(
        kind='bar',
        stacked=True,
        color=[JOURNAL_COLORS[j] for j in topic_journal_normalized.columns],
        width=0.8
    )

    plt.title('Journal distribution by topic', pad=20)
    plt.xlabel('Topic', labelpad=10)
    plt.ylabel('Document share', labelpad=10)
    plt.legend(
        title='Journal',
        bbox_to_anchor=(1.05, 1),
        loc='upper left',
        borderaxespad=0.
    )
    plt.xticks(rotation=0, ha='center')
    plt.tight_layout()
    plt.show()

def visualize_umap(docs_meta, theta, journal_colors=JOURNAL_COLORS):
    """интерактивная визуализация документов"""
    journals = [doc['journal'] for doc in docs_meta]
    doc_ids = [doc['id'] for doc in docs_meta]
    doc_texts = [doc['text'][:150] + '...' for doc in docs_meta]

    reducer = umap.UMAP(min_dist=0.1, n_neighbors=50, metric='euclidean')
    embedding = reducer.fit_transform(theta)

    df = pd.DataFrame({
        'x': embedding[:, 0],
        'y': embedding[:, 1],
        'journal': journals,
        'id': doc_ids,
        'text_preview': doc_texts,
        'color': [journal_colors[j] for j in journals]
    })

    fig = px.scatter(
```

```
        df,
        x='x',
        y='y',
        color='journal',
        color_discrete_map=journal_colors,
        hover_data={
            'x': False,
            'y': False,
            'journal': True,
            'id': True,
            'text_preview': True
        },
        labels={
            'journal': 'Journal',
            'id': 'Document ID',
            'text_preview': 'Text preview'
        },
        title='Interactive document visualization by journal (UMAP)'
    )

    fig.update_layout(
        plot_bgcolor='white',
        xaxis_title='UMAP 1',
        yaxis_title='UMAP 2',
        hovermode='closest',
        showlegend=True
    )

    fig.update_xaxes(showgrid=False)
    fig.update_yaxes(showgrid=False)

    output_path = Path("umap_visualization.html")
    plot(fig, filename=str(output_path), auto_open=False)
    print(f"Visualization saved to: {output_path.absolute()}")
    webbrowser.open(f'file://{output_path.absolute()}')

def train_tomotopy_model(tokenized_docs, num_topics=NUM_TOPICS):
    """обучение tomotopy lda модели"""
    model = tomotopy.LDAModel(
        k=num_topics,
        alpha=ALPHA,
        eta=BETA,
        min_cf=MIN_CF,
        min_df=MIN_DF,
        rm_top=RM_TOP,
        seed=RANDOM_STATE
    )

    for doc in tokenized_docs:
        model.add_doc(doc)

    model.train(ITERATIONS)

    return model

def print_top_words(model, num_words=TOP_K):
    """вывод топ-слов по темам"""
    for k in range(model.k):
        words = model.get_topic_words(k, top_n=num_words)
        print(f"Topic {k}: {', '.join(word for word, _ in words)}")

def get_topic_document_distribution(model):
    """распределение тем по документам"""
    return np.array([doc.get_topic_dist() for doc in model.docs])

def calculate_coherence(model, tokenized_docs):
    """расчет когерентности для tomotopy"""
    coherence = tomotopy.coherence.Coherence(model, coherence='u_mass')
```

```
    return coherence.get_score()

def load_data(input_path):
    """загрузка данных и разделение по журналам"""
    with open(input_path, encoding='utf-8') as f:
        docs = json.load(f)

    journal_docs = {
        'ALL': [doc for doc in docs if doc.get('text')],
        'DHQ': [doc for doc in docs if doc.get('journal') == 'DHQ' and doc.get('text')],
        'DSH': [doc for doc in docs if doc.get('journal') == 'DSH' and doc.get('text')],
        'LChN': [doc for doc in docs if doc.get('journal') == 'LChN' and
doc.get('text')],
        'JCA': [doc for doc in docs if doc.get('journal') == 'JCA' and doc.get('text')],
        'IJDH': [doc for doc in docs if doc.get('journal') == 'IJDH' and doc.get('text')]
    }

    return docs, journal_docs

def preprocess_documents(documents):
    """токенизация документов"""
    return [doc['text'].split() for doc in documents]

def save_tomotopy_metrics(metrics, output_dir):
    """сохранение всех метрик tomotopy в json файл"""
    output_dir.mkdir(parents=True, exist_ok=True)

    def serialize(obj):
        if isinstance(obj, (np.float32, np.float64)):
            return float(obj)
        if isinstance(obj, np.ndarray):
            return obj.tolist()
        return obj

    serialized_metrics = {k: serialize(v) for k, v in metrics.items()}

    with open(output_dir / 'tomotopy_metrics.json', 'w', encoding='utf-8') as f:
        json.dump(serialized_metrics, f, indent=2, ensure_ascii=False)
```

Модуль тематических моделей надстроек над LDA

```
from pathlib import Path
import json
from collections import defaultdict
import pandas as pd
import tomotopy
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings

class Config:
    # параметры DTM
    TIME_FIELD = "year"
    MIN_TIME = 2016
    MAX_TIME = 2025
    TIME_STEP = 3
    NUM_TOPICS = 15
    ALPHA_VAR = 0.01
    ETA_VAR = 0.001
    PHI_VAR = 0.1

    # learning rate = a / (b + i)^c
    LR_A = 0.1
    LR_B = 1
    LR_C = 0.75

    # параметры hLDA
```

```
    GAMMA = 1
    DEPTH = 4
    ALPHA = 0.1
    BETA = 0.01



    # общие параметры
    TOP_K = 10
    ITERATIONS = 100

    MIN_CF = 3
    MIN_DF = 3
    RM_TOP = 10

    RANDOM_STATE = 20
    INPUT_JSON_PATH = Path("preprocessed/Test_T_ALL.json")
    OUTPUT_DIR = Path("tm_output")

def load_data(config, grouped=False):
    """Загрузка и подготовка данных"""
    with open(config.INPUT_JSON_PATH, encoding='utf-8') as f:
        data = json.load(f)

    if grouped:
        yearly_docs = defaultdict(list)
        for doc in data:
            year = doc.get(config.TIME_FIELD)
            if year and config.MIN_TIME <= int(year) <= config.MAX_TIME:
                yearly_docs[int(year)].append(doc["text"])

        year_to_group, year_groups = group_years(yearly_docs.keys(), config.TIME_STEP)
        grouped_docs = defaultdict(list)
        for year, docs in yearly_docs.items():
            group = year_to_group[year]
            grouped_docs[group].extend(docs)

        time_intervals = []
        for group_start, group_end in year_groups:
            group_name = f"{group_start}-{group_end}"
            if group_name in grouped_docs:
                time_intervals.append(group_name)

        return grouped_docs, time_intervals, year_groups

    else:
        docs_by_year = defaultdict(list)
        for doc in data:
            year = doc.get(config.TIME_FIELD)
            if year and config.MIN_TIME <= int(year) <= config.MAX_TIME:
                docs_by_year[int(year)].append(doc["text"])

        valid_years = sorted(docs_by_year.keys())
        expected_years = set(range(min(valid_years), max(valid_years)+1))

        if expected_years != set(valid_years):
            missing = sorted(expected_years - set(valid_years))
            raise ValueError(f"Пропущенные годы в данных: {missing}")

        return docs_by_year, valid_years

def group_years(years, step):
    """Группировка годов по заданному шагу"""
    min_year = min(years)
    max_year = max(years)

    groups = []
    current = min_year
```

```
    while current <= max_year:
        group_end = current + step - 1
        groups.append((current, group_end))
        current = group_end + 1

    year_to_group = {}
    for year in years:
        for group_start, group_end in groups:
            if group_start <= year <= group_end:
                year_to_group[year] = f"{group_start}-{group_end}"
                break

    return year_to_group, groups

def train_dtm_model(docs_by_year, valid_years, config):
    """Обучение DTM модели"""
    model = tomotopy.DTModel(
        k=config.NUM_TOPICS,
        t=len(valid_years),
        min_cf=config.MIN_CF,
        min_df=config.MIN_DF,
        rm_top=config.RM_TOP,
        alpha_var=config.ALPHA_VAR,
        eta_var=config.ETA_VAR,
        phi_var=PHI_VAR,
        lr_a=LR_A,
        lr_b=LR_B,
        lr_c=LR_C,
        seed=config.RANDOM_STATE
    )

    year_to_idx = {y: i for i, y in enumerate(valid_years)}

    for year, docs in docs_by_year.items():
        for doc in docs:
            model.add_doc(doc.strip().split(), timepoint=year_to_idx[year])

    for i in range(0, config.ITERATIONS, 10):
        model.train(10)

    return model, year_to_idx

def train_hlda_model(tokenized_docs, config):
    """Обучение hLDA модели"""
    model = tomotopy.HLDAModel(
        tw=tomotopy.TermWeight.ONE,
        depth=config.DEPTH,
        min_cf=config.MIN_CF,
        rm_top=config.RM_TOP,
        gamma=config.GAMMA,
        alpha=config.ALPHA,
        eta=config.BETA,
        seed=config.RANDOM_STATE
    )

    for doc in tokenized_docs:
        model.add_doc(doc)

    for i in range(config.ITERATIONS):
        model.train(1)

    return model

def compute_topic_diversity(model, topic_id, timepoint, top_k):
    """Вычисление разнообразия тем"""
    word_probs = model.get_topic_word_dist(topic_id, timepoint)
    top_word_indices = np.argsort(word_probs)[::-1][:top_k]
    return set(top_word_indices)
```

```
def calculate_global_diversity(model, time_points, config):
    """Вычисление глобального разнообразия тем"""
    all_unique_words = set()
    total_top_words = 0

    for topic_id in range(config.NUM_TOPICS):
        for time_idx in range(len(time_points)):
            top_words = compute_topic_diversity(model, topic_id, time_idx, config.TOP_K)
            all_unique_words.update(top_words)
            total_top_words += config.TOP_K

    return len(all_unique_words) / total_top_words if total_top_words > 0 else 0.0

def save_topic_dynamics(model, time_points, config, output_path, year_groups=None):
    """Сохранение динамики тем в файл"""
    output = []
    global_diversity = calculate_global_diversity(model, time_points, config)

    for topic_id in range(config.NUM_TOPICS):
        topic_data = {
            "topic_id": topic_id,
            "dynamics": []
        }

        topic_unique_words = set()
        topic_total_words = 0

        for time_idx, interval in enumerate(time_points):
            words       =       model.get_topic_words(topic_id,       timepoint=time_idx,
top_n=config.TOP_K)
            top_words = compute_topic_diversity(model, topic_id, time_idx, config.TOP_K)
            yearly_diversity = len(top_words) / config.TOP_K

            if year_groups and '-' in interval:
                start_year, end_year = map(int, interval.split('-'))
                time_data = {
                    "time_interval": interval,
                    "start_year": start_year,
                    "end_year": end_year,
                    "diversity": yearly_diversity,
                    "top_words": [{"word": w, "prob": float(p)} for w, p in words]
                }
            else:
                time_data = {
                    "year": int(interval),
                    "diversity": yearly_diversity,
                    "top_words": [{"word": w, "prob": float(p)} for w, p in words]
                }

            topic_data["dynamics"].append(time_data)
            topic_unique_words.update(top_words)
            topic_total_words += config.TOP_K

        topic_data["topic_diversity"] = len(topic_unique_words) / topic_total_words if
topic_total_words > 0 else 0
        output.append(topic_data)

    result = {
        "global_diversity": global_diversity,
        "topics": output
    }

    if year_groups:
        result["time_step"] = config.TIME_STEP
        result["year_groups"] = [{"start": s, "end": e} for s, e in year_groups]

    config.OUTPUT_DIR.mkdir(parents=True, exist_ok=True)
```

```
    with open(output_path, 'w', encoding='utf-8') as f:
        json.dump(result, f, indent=2, ensure_ascii=False)

def save_hierarchy(model, output_dir, top_k=10):
    """Сохранение иерархии тем hLDA"""
    hierarchy = []
    for k in range(model.k):
        level = model.level(k)
        parent = model.parent_topic(k)
        words = [word for word, _ in model.get_topic_words(k, top_n=top_k)]
        hierarchy.append({
            "topic_id": k,
            "level": level,
            "parent": parent,
            "top_words": words
        })

    with open(output_dir / "hierarchy.json", "w", encoding="utf-8") as f:
        json.dump(hierarchy, f, indent=2, ensure_ascii=False)

    with open(output_dir / "hierarchy.txt", "w", encoding="utf-8") as f:
        for topic in hierarchy:
            indent = "  " * topic["level"]
            parent_info = f", Parent {topic['parent']}" if topic["level"] > 0 else ""
            f.write(
                f"{indent}Topic                {topic['topic_id']}                (Level
{topic['level']}{parent_info}): "
                f"{', '.join(topic['top_words'])}\n"
            )

def visualize_word_dynamics(model, time_points, topic_id, words_to_track):
    """Визуализация динамики слов в теме"""
    data = []

    for time_idx, interval in enumerate(time_points):
        topic_words = model.get_topic_words(topic_id, timepoint=time_idx, top_n=50)
        word_probs = {w: p for w, p in topic_words}

        for word in words_to_track:
            if word in word_probs:
                if '-' in interval:
                    start, end = map(int, interval.split('-'))
                    mid_year = (start + end) / 2
                    x_val = mid_year
                    x_label = interval
                else:
                    x_val = int(interval)
                    x_label = interval

                data.append({
                    'Interval': interval,
                    'XValue': x_val,
                    'Word': word,
                    'Probability': word_probs[word]
                })

    if not data:
        return

    df = pd.DataFrame(data)
    sns.set_style("white")
    plt.figure(figsize=(12, 6))

    ax = sns.lineplot(
        data=df,
        x='XValue',
        y='Probability',
```

```
        hue='Word',
        marker='o',
        linewidth=2.5,
        markersize=10,
        palette="tab10"
    )

    ax.set_title(f'Динамика вероятности слов в теме {topic_id}', fontsize=16)
    ax.set_xlabel('Год', fontsize=14)
    ax.set_ylabel('Вероятность слова', fontsize=14)

    intervals = sorted(df['Interval'].unique())
    x_vals = sorted(df['XValue'].unique())
    ax.set_xticks(x_vals)
    ax.set_xticklabels(intervals)

    ax.tick_params(axis='both', which='major', labelsize=12)
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)

    plt.legend(title='Слова', frameon=False)
    plt.tight_layout()
    plt.show()

def run_dtm_pipeline(config):
    """Пайплайн для DTM модели"""
    try:
        docs_by_year, valid_years = load_data(config)
        model, year_to_idx = train_dtm_model(docs_by_year, valid_years, config)

        output_dir = config.OUTPUT_DIR / "DTM"
        output_dir.mkdir(parents=True, exist_ok=True)

        save_topic_dynamics(model, valid_years, config, output_dir /
"topic_dynamics.json")

        visualize_word_dynamics(
            model,
            valid_years,
            topic_id=0,
            words_to_track=['data', 'model', 'learning']
        )

    except Exception as e:
        print(f"Ошибка: {str(e)}")

def run_hlda_pipeline(config):
    """Пайплайн для hLDA модели"""
    try:
        with open(config.INPUT_JSON_PATH, encoding='utf-8') as f:
            data = json.load(f)

        tokenized_docs = [doc["text"].split() for doc in data if doc.get("text")]

        model = train_hlda_model(tokenized_docs, config)

        output_dir = config.OUTPUT_DIR / "hLDA"
        output_dir.mkdir(parents=True, exist_ok=True)

        save_hierarchy(model, output_dir)
        visualize_hierarchy(model, output_dir)

        metrics = {
            "perplexity": model.perplexity,
            "log_likelihood": model.ll_per_word,
        }

        with open(output_dir / "metrics.json", "w", encoding="utf-8") as f:
```

```
            json.dump(metrics, f, indent=2)

    except Exception as e:
        print(f"Ошибка: {str(e)}")

config = Config()
run_dtm_pipeline(config)
run_hlda_pipeline(config)
```

Модуль ARTM

```
import artm
import json
from pathlib import Path
from collections import Counter
import pandas as pd

# константы
NUM_TOPICS = 30
NUM_PASSES = 50
TOP_TOKENS_NUM = 12
SAVE_PHI_MATRIX = False
INPUT_JSON = "Test_T_ALL.json"
VW_FILENAME = "topic_modeling_input.vw"
PHI_MATRIX_FILENAME = "phi_matrix.csv"

class ARTM_Model:
    def __init__(self, use_regularizers=False):
        """
        инициализация модели ARTM
        use_regularizers: флаг использования регуляризаторов
        """
        self.use_regularizers = use_regularizers
        self.model = None
        self.batch_vectorizer = None

    def prepare_data(self, input_json):
        """
        преобразование json в vw формат
        input_json: путь к входному json файлу
        возвращает путь к созданному vw файлу
        """
        with open(input_json, encoding="utf-8") as f:
            raw_docs = json.load(f)

        docs = [doc["text"].split() for doc in raw_docs if doc.get("text")]

        vw_lines = []
        for i, doc_tokens in enumerate(docs):
            doc_id = f"doc{i}"
            token_counts = Counter(doc_tokens)
            vw_content = " ".join(f"{term}:{count}" for term, count in
token_counts.items())
            vw_lines.append(f"{doc_id} |text {vw_content}")

        vw_path = Path(VW_FILENAME)
        vw_path.parent.mkdir(parents=True, exist_ok=True)

        with open(vw_path, "w", encoding="utf-8") as f:
            f.write("\n".join(vw_lines))

        print(f"подготовлено документов: {len(vw_lines)}")
        return vw_path

    def save_phi_matrix(self, filename=PHI_MATRIX_FILENAME):
        """
        сохранение матрицы phi в csv файл
```

```
        filename: имя выходного файла
        """
        phi = self.model.get_phi()
        df_phi = pd.DataFrame(phi)
        df_phi.to_csv(filename)
        print(f"матрица phi сохранена в {filename}")

    def run_experiment(self, vw_path):
        """
        обучение модели ARTM
        vw_path: путь к vw файлу с данными
        """
        batches_dir = Path("batches") / vw_path.stem
        batches_dir.mkdir(parents=True, exist_ok=True)

        self.batch_vectorizer = artm.BatchVectorizer(
            data_path=str(vw_path),
            data_format='vowpal_wabbit',
            target_folder=str(batches_dir),
            batch_size=100
        )

        dictionary = artm.Dictionary()
        dictionary.gather(data_path=str(batches_dir))

        self.model = artm.ARTM(
            topic_names=[f'topic_{i}' for i in range(NUM_TOPICS)],
            dictionary=dictionary,
            cache_theta=False
        )

        # добавление метрик
        self.model.scores.add(artm.PerplexityScore(name='perplexity',
dictionary=dictionary))
        self.model.scores.add(artm.SparsityPhiScore(name='sparsity_phi'))
        self.model.scores.add(artm.SparsityThetaScore(name='sparsity_theta'))
        self.model.scores.add(artm.TopTokensScore(
            name='top_tokens',
            num_tokens=TOP_TOKENS_NUM,
            class_id='text'
        ))

        # добавление регуляризаторов при необходимости
        if self.use_regularizers:

self.model.regularizers.add(artm.SmoothSparseThetaRegularizer(name='sparse_theta',  tau=-
1))

self.model.regularizers.add(artm.SmoothSparsePhiRegularizer(name='sparse_phi', tau=-1))

self.model.regularizers.add(artm.DecorrelatorPhiRegularizer(name='decorrelator_phi',
tau=2.5e5))

        # обучение модели
        self.model.num_document_passes = 1
        self.model.fit_offline(
            batch_vectorizer=self.batch_vectorizer,
            num_collection_passes=NUM_PASSES)

        if SAVE_PHI_MATRIX:
            self.save_phi_matrix()

    def print_results(self):
        """
        вывод результатов обучения
        """
        perplexity = self.model.score_tracker['perplexity'].last_value
        print(f"\nперплексия модели: {perplexity:.2f}")
```

```
        print("разреженность phi:", self.model.score_tracker['sparsity_phi'].last_value)
        print("разреженность                                                    theta:",
self.model.score_tracker['sparsity_theta'].last_value)

        print("\nтоповые токены по темам:")
        top_tokens = self.model.score_tracker['top_tokens'].last_tokens
        for topic_name in self.model.topic_names:
            print(f"{topic_name}: {', '.join(top_tokens[topic_name])}")

def run_plsa():
    """
    запуск модели pLSA (ARTM без регуляризаторов)
    """
    model = ARTM_Model(use_regularizers=False)
    vw_path = model.prepare_data(INPUT_JSON)
    model.run_experiment(vw_path)
    model.print_results()

def run_artm():
    """
    запуск модели ARTM с регуляризаторами
    """
    model = ARTM_Model(use_regularizers=True)
    vw_path = model.prepare_data(INPUT_JSON)
    model.run_experiment(vw_path)
    model.print_results()

run_plsa() # запуск pLSA
run_artm() # запуск ARTM
```

Модуль предобработки и подсчёта статистик

```
class TextPreprocessor:
    """
    модуль предобработки текстовых данных
    """
    # константы путей
    INPUT_JSON = "Test.json"
    STOPWORDS_PATH = "stopwords.txt"
    OUTPUT_DIR = "preprocessed"

    # параметры обработки
    MODES = {
        1: "T",     # токенизация
        2: "TS",    # токенизация + стоп-слова
        3: "TL",    # токенизация + лемматизация
        4: "TSL"    # всё вместе
    }

    POS_FILTERS = {
        1: {"NOUN", "VERB"}, # существительные и глаголы
        2: {"NOUN"}, # существительные
        3: None  # без фильтра
    }

    def __init__(self, mode=4, pos_mode=3, remove_names=False):
        self.mode = mode
        self.pos_mode = pos_mode
        self.remove_names = remove_names
        self.nlp = spacy.load("en_core_web_sm", disable=["parser"])
        self.nlp.max_length = 2_000_000
        self.custom_stopwords = self._load_stopwords()
        self.special_cases = {"datum": "data"}

    def _load_stopwords(self):
        """загрузка стоп-слов из файла"""
        with open(self.STOPWORDS_PATH, encoding="utf-8") as f:
```

```
            return {line.strip().lower() for line in f if line.strip()}

    def _clean_text(self, text):
        """базовая очистка текста"""
        text = text.lower()
        text = re.sub(r'[^\x00-\x7F]+', ' ', text)  # удаление unicode
        text = re.sub(r'[^a-z\s]', ' ', text)       # удаление пунктуации/цифр
        return re.sub(r'\s+', ' ', text).strip()

    def preprocess_text(self, text):
        """основная функция предобработки текста"""
        text = self._clean_text(text)
        doc = self.nlp(text)
        allowed_pos = self.POS_FILTERS[self.pos_mode]
        result = []

        for token in doc:
            if not token.is_alpha:
                continue
            if allowed_pos and token.pos_ not in allowed_pos:
                continue

            word = token.text if self.mode in (1, 2) else token.lemma_
            word = word.lower()

            if len(word) == 1:
                continue
            if self.mode in (2, 4) and word in self.custom_stopwords:
                continue
            if word in self.special_cases:
                word = self.special_cases[word]

            result.append(word)

        return ' '.join(result)

    def process_corpus(self):
        """обработка всего корпуса"""
        with open(self.INPUT_JSON, encoding="utf-8") as f:
            data = json.load(f)

        processed = []
        for item in data:
            if not item.get("text", "").strip():
                continue

            clean_text = self.preprocess_text(item["text"])
            new_item = item.copy()
            new_item["text"] = clean_text
            processed.append(new_item)

        output_name = f"Test_{self.MODES[self.mode]}_{self.pos_mode}.json"
        os.makedirs(self.OUTPUT_DIR, exist_ok=True)

        with open(Path(self.OUTPUT_DIR)/output_name, "w", encoding="utf-8") as f:
            json.dump(processed, f, ensure_ascii=False, indent=2)

        return f"processed {len(processed)} documents, saved in {output_name}"

class CorpusAnalyzer:
    """
    модуль анализа текстового корпуса
    """

    # константы
    INPUT_JSON = "preprocessed/Test_T_ALL.json"
    STATS_DIR = "stats"
```

```
    def __init__(self, ngram_size=2, top_k=1000, min_df=5):
        self.ngram_size = ngram_size
        self.top_k = top_k
        self.min_df = min_df
        os.makedirs(self.STATS_DIR, exist_ok=True)

    def _load_corpus(self):
        """загрузка корпуса из json"""
        with open(self.INPUT_JSON, encoding="utf-8") as f:
            return json.load(f)

    def get_frequent_ngrams(self):
        """вычисление частотных n-грамм"""
        data = self._load_corpus()
        all_tokens = []

        for item in data:
            all_tokens.extend(item["text"].split())

        ngrams = zip(*[all_tokens[i:] for i in range(self.ngram_size)])
        counter = Counter(ngrams)

        output_path                                                                    =
Path(self.STATS_DIR)/f"Test_{self.ngram_size}_grams_top_{self.top_k}.csv"
        pd.DataFrame(
            [(" ".join(k), v) for k, v in counter.most_common(self.top_k)],
            columns=["ngram", "count"]
        ).to_csv(output_path, index=False)

        return f"топ {self.top_k} {self.ngram_size}-грамм сохранен в {output_path}"

    def get_tfidf_ngrams(self):
        """вычисление tf-idf для n-грамм"""
        data = self._load_corpus()
        texts = [item["text"] for item in data]

        vectorizer = TfidfVectorizer(
            ngram_range=(self.ngram_size, self.ngram_size),
            min_df=self.min_df
        )
        tfidf_matrix = vectorizer.fit_transform(texts)

        features = vectorizer.get_feature_names_out()
        scores = tfidf_matrix.sum(axis=0).A1
        sorted_indices = scores.argsort()[::-1][:self.top_k]

        output_path                                                                    =
Path(self.STATS_DIR)/f"Test_tfidf_{self.ngram_size}_grams_{self.top_k}.csv"
        pd.DataFrame({
            "ngram": features[sorted_indices],
            "tfidf": scores[sorted_indices]
        }).to_csv(output_path, index=False)

        return f"tf-idf for {self.ngram_size}-gram saved in {output_path}"

preprocessor = TextPreprocessor(mode=4, pos_mode=3)

analyzer = CorpusAnalyzer(ngram_size=2, top_k=1000)
print(analyzer.get_frequent_ngrams())
print(analyzer.get_tfidf_ngrams())
```