**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**
**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**
**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**
**Кафедра биомедицинской информатики**

СИНЯВСКИЙ
Тимур Владимирович

# ПОСТРОЕНИЕ СЕМАНТИЧЕСКИХ ВЕКТОРНЫХ ПРЕДСТАВЛЕНИЙ ТЕКСТОВЫХ ДОКУМЕНТОВ

Дипломная работа

Научный руководитель:
ст. преподаватель кафедры БМИ
Николаев Г. И.

Научный консультант:
к.т.н., ОИПИ НАН Беларуси
Гецевич Ю.С.

Допущен к защите
«___» _____ 2021 г.
Зав. кафедрой биомедицинской информатики
доцент Ю.Л. Орлович

Минск, 2021

# РЕФЕРАТ

Дипломная работа: 43 страницы, 20 рисунков, 3 таблицы, 15 источников, 2 приложения.

Ключевые слова: ЭМБЕДДИНГИ, ТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ, КЛАСТЕРИЗАЦИЯ, КЛЮЧЕВЫЕ СЛОВА.

Объект исследования: текстовая информация.

Цель работы: реализация пайплайна тематического моделирования для данных белорусских источников.

Результат: проведено сравнение различных алгоритмов, из них выбран оптимальный, разработана система сбора статей и применения пайплайна тематического моделирования, а также сайт для визуализаций результатов.

Область применения: новостные агрегаторы, разовый анализ любого корпуса, анализ трендов.

# РЭФЕРАТ

Дыпломная праца: 43 старонкі, 20 малюнка, 3 табліцы, 15 крыніц, 2 дадатка.

Ключавыя словы: ЭМБЭДДІНГІ, ТЭМАТЫЧНАЕ МАДЭЛЯВАННЕ, КЛАСТАРЫЗАЦЫЯ, КЛЮЧАВЫЯ СЛОВЫ.

Аб’ект даследавання: тэкставая інфармацыя.

Мэта работы: рэалізацыя пайплайна тэматычнага мадэлявання для дадзеных беларускіх крыніц.

Вынік: праведзена параўнанне розных алгарытмаў, з іх абраны аптымальны, распрацавана сістэма збору артыкулаў і прымянення пайплайна тэматычнага мадэлявання, а таксама сайт для візуалізацыі вынікаў.

Вобласць прымянення: навінавыя агрэгатары, разавы аналіз любога корпуса, аналіз трэндаў.

# ABSTRACT

Diploma thesis: 43 pages, 20 figures, 3 tables, 15 sources, 2 applications.

Keywords: EMBEDDINGS, TOPIC MODELING, CLUSTERING, KEYWORDS.

Object of research: text data.

Objective: implementation of a topic modeling pipeline for data from belarusian sources.

Result: a comparison of various algorithms was conducted and the optimal one was selected, a system for collecting articles and applying a topic modeling pipeline was developed, as well as a site for results visualization.

Scope: news aggregators, one-time analysis of any corpus, trend analysis.

# ОГЛАВЛЕНИЕ

# ПЕРЕЧЕНЬ ТЕРМИНОВ

Корпус (corpus) - набор текстов.
Корпора (corpora) - множество корпусов.
Документ (document) - текст, элемент корпуса.
Словарь (vocabulary) - все слова (токены) корпуса.
Эмбеддинг (embedding) - векторное представление слова или документа.

# ВВЕДЕНИЕ

В настоящее время объемы текстовых данных постоянно растут, в частности благодаря средствам массовой информации и соцсетям, также сюда можно включить сайты научных публикаций, такие как arxiv или pubmed. В результате часто бывает очень сложно найти релевантную информацию из такого массива данных. Машинное обучение может помочь обобщить и структурировать большой объем текстов и извлечь из него информацию. В частности для этого могут использоваться эмбеддинги и тематическое моделирование.

Эмбеддинги слов и документов являются одними из ключевых понятий в обработке текста. Их можно использовать для:

- Поиска документов по текстовому запросу.
- Поиска похожих документов на заданный документ.
- Группировки схожих документов.
- Как входные данные для других моделей обработки текста, например классификаторов текста.

Тематическое моделирование помогает обнаруживать семантическую структуру (темы) присутствующую в корпусе. Также это может помочь, например, определить тренды тем за определенный промежуток времени.

На данный момент один из последних алгоритмов тематического моделирования - top2vec [10]. Автором показано, что он работает лучше, чем классические алгоритмы для данной задачи (LDA и PLSA). Top2vec использует эмбеддинги doc2vec, но существуют и другие алгоритмы построения эмбеддингов документов, которые могут быть использованы вместо doc2vec. В данной работе проведено сравнение результатов алгоритма с различными эмбеддингами на данных белорусских источников и выбран лучший вариант

для дальнейшего использования. Изменение алгоритма эмбеддингов потребовало также изменить алгоритм нахождения ключевых слов. Важной особенностью данного пайплайна тематического моделирования является то, что в алгоритмах используются только тексты без каких-либо меток (unsupervised подход), это значит, что он может быть применен к любому корпусу текстов на русском языке, а не только представленному в данной работе (для языка специфичен только шаг предобработки текста, остальные части не зависят от языка).

Также разработана система для сбора и агрегации статей и документов из различных постоянно обновляющихся источников и для обработки полученных данных построенным пайплайном. В дальнейшем эти данные могут использоваться каким-либо приложением с пользовательским интерфейсом, например новостным агрегатором, или для визуализаций и аналитики полученных данных. Пример такого приложения также реализован в работе.

Актуальность работы обуславливаются важностью и сложностью анализа и фильтрации информации в настоящее время. Алгоритм в работе позволяет получить разбиение на темы, которое лучше соответствует конкретному корпусу, так как темы определяются исходя из данных, а не фиксируются заранее, как это делается в классических новостных агрегаторах (например Google news). Также в разработанном решении демонстрируется работа поиска схожих артиклей и отображение трендов для белорусских СМИ (на данный момент экспериментально для нескольких источников, но это масштабируемо и для большего их количества). Отечественных аналогов таких инструментов на момент написания не найдено.

# Глава 1. ОБЗОР МЕТОДОВ

## 1.1 Предобработка текста

Предобработка текста зачастую является первым шагом в пайплайнах обработки теста. Предварительная очистка текста способна существенно улучшить качество алгоритмов, которые потребляют текст, например алгоритмов построения эмбеддингов. Предобработка текста может включать в себя различные шаги. Набор и последовательность этих шагов зависит от имеющихся данных и от задачи, которую требуется решить. Так некоторая предобработка может улучшить результаты в одной задаче и ухудшить в другой.

Возможные варианты этапов предобработки текста:

- Удаление цифр, знаков препинания, ссылок или других ненужных символов.
- Приведение всего теста в нижний регистр. Помогает убрать некоторые дубликаты, например “белок” и “Белок”.
- Токенизация (tokenization) - разбиение текста на токены. Обычно токенами являются слова, но также могут быть слоги, числа и знаки.
- Удаление стоп-слов. Стоп-слова - это слова, которые часто встречаются в языке и не несут сильной смысловой нагрузки, например, предлоги и местоимения. Также стоп-слова могут быть специфичны для заданного корпуса и определяться автоматически, например с помощью подсчета IDF (inverse document frequency).
- Стемминг (stemming) - приведение слов к стандартной форме путем отсечения суффиксов и префиксов. Например, токены “корова” и “коровами” приводится к токену “коров”.
- Лемматизация (lemmatization) - как и стемминг, это приведение слова к стандартной форме, но используя морфологический анализ. В этом случае слова “корова” и “коровами” оба приводятся к слову “корова”.

Конечно это не полный список возможных шагов предобработки текста, но одни из самых часто используемых.

## 1.2 Построение эмбеддингов

Построение эмбеддингов - широко распространённый метод в машинном обучении. Он может применяться к различным типам данных: изображениям, аудио, текстам, графам и так далее. Суть построения эмбеддингов заключается в отображении элементов (например слов) в вектора, которые в дальнейшем могут использоваться для анализа или как входные данные для других алгоритмов (например классификации). В случае текстов эмбеддинги можно получать как для отдельных слов, так и для предложений и целых документов.

### 1.2.1 Алгоритмы построения эмбеддингов слов

One-hot encoding - один из самых простых методов построения эмбеддингов слов. Он строит бинарные вектора размерности равной длине словаря. Ставит единицу в позиции равной индексу кодируемого слова в словаре и нули во всех остальных позициях. Одна из проблем этого метода - очень большая размерность векторов. Другая - не сохраняется связь между словами, то есть проблемно ввести меру сходства слов.

Word2vec [1] - алгоритм построения эмбеддингов, который решает обе предыдущие проблемы. Он опирается на дистрибутивную гипотезу: “слова, встречающиеся в схожих контекстах, имеют близкие значения”. Существует две версии алгоритма: skip-gram и continuous bag-of-words (CBOW). Идея метода состоит в том, чтобы строить эмбеддинги путём оптимизации задачи предсказания контекстных слов по заданному (skip-gram версия) или наоборот (CBOW). То есть в случае модели skip-gram берётся скользящее окно по тексту, в окне берётся центральное слово и по нему предсказываются вероятности появления всех остальных слов в окне. Задача оптимизации - максимизировать вероятности контекстных слов, а оптимизационные параметры и есть эмбеддинги слов. Оптимизация выполняется алгоритмом градиентного спуска. В случае модели CBOW всё аналогично, только центральное слово в окне предсказывается по сумме контекстных слов.

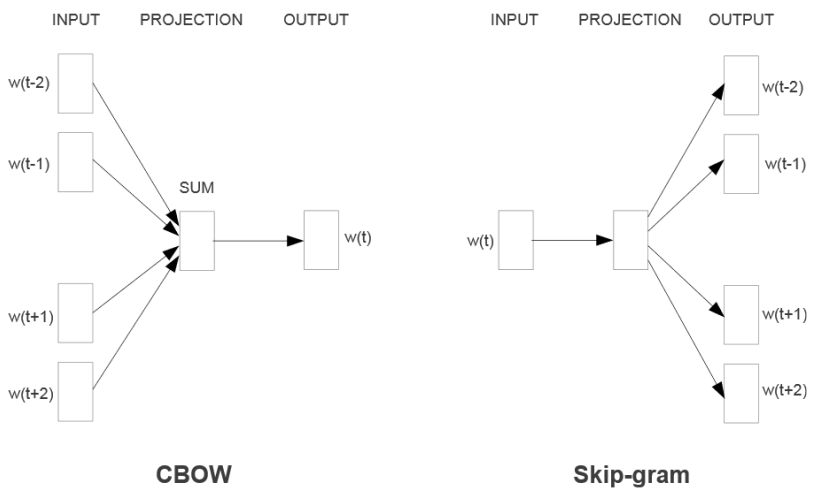



**Рисунок 1.1 - Разновидности word2vec**

Global Vectors (GloVe) [2] - ещё один похожий метод. Он использует матрицу совместной встречаемости слов в корпусе и максимизирует скалярное произведение векторов слов, которые встречаются вместе чаще.

Часто эмбеддинги слов обладают двумя полезными свойствами. Первое: близкие по значению слова близки по метрике (чаще всего косинусной). То есть, если взять ближайшие слова к заданному, то можно получить его синонимы. Второе свойство: эмбеддинги сохраняют семантические отношения в структуре векторного пространства. Распространенный пример этого свойства это выражение “man - woman = king - queen”. То есть если подставить вместо слов эмбеддинги, то справа и слева будут близкие значения, показывающие отношение женщина-мужчина.

### 1.2.2 Алгоритмы построения эмбеддингов документов

Bag-of-words - один из самых примитивных методов построения эмбеддингов документов. По сути в данной модели эмбеддинг документа равен сумме one-hot эмбеддингов слов, которые документ содержит. То есть это вектор размерности словаря корпуса, в котором на k-й позиции стоит число вхождений в документ слова, индекс в словаре которого равен k. Такой эмбеддинг теряет информацию о порядке слов, отсюда и название.

Bag-of-n-grams - обобщение метода bag-of-words, в котором элементами словаря являются не отдельные слова, а последовательности слов стоящих рядом, то есть n-gram. Такой подход позволяет лучше учитывать порядок слов, но зато еще больше увеличивает размерность эмбеддинга.

Term frequency - inverse document frequency (TF-IDF) - ещё один алгоритм, в котором эмбеддинг имеет размерность равную длине словаря. Для каждой пары документ-слово рассчитывается величина

$$tfidf(w, d) = tf(w, d) * log(\frac{N}{df(w)}),$$

где tf(w, d) - число вхождений слова w в документ d, df(w) - число документов содержащих слово w, N - общее число документов в корпусе. Эмбеддинг это набор таких значений рассчитанных для данного документа со всеми словами словаря. Если взять только первый множитель выражения, то получится метод bag-of-words. Второй множитель позволяет акцентировать внимание на более редких словах и меньше учитывать слова, которые есть в большинстве документов.

Агрегация эмбеддингов слов - подход, в котором для всех слов в документе строятся эмбеддинги каким-либо методом, например word2vec, а потом к ним применяется функция, например суммы или среднего, после чего получается эмбеддинг документа. В качестве функции агрегации также можно взять взвешенную сумму эмбеддингов слов, где весами будут значения idf описанные ранее.

Doc2vec [4] является обобщением алгоритма word2vec для документов. Он так же как и word2vec имеет две версии: distributed bag of words (DBOW) и distributed memory (DM). Версия DM базируется на CBOW версии word2vec. На входе модели к векторам контекстных слов добавляется вектор документа, из которого выбраны контекстные слова. Изначально для каждого документа вектор инициализируется случайно, как и для слов. Оптимизация выполняется также градиентным спуском. Оптимизационные параметры - эмбеддинги документов и слов. Эмбеддинги слов могут быть обучены отдельно моделью word2vec, в таком случае в doc2vec оптимизируются только эмбеддинги документов. Для получения эмбеддинга нового документа после обучения модели проводится несколько итераций оптимизации с фиксированными эмбеддингами слов. То есть эмбеддинг документа модифицируется так, чтобы давать максимальные вероятности слов, которые он содержит. Версия DBOW doc2vec основана на skip-gram версии word2vec. В ней вместо входного вектора слова берется вектор документа и по нему предсказывается вероятность слов в этом документе. В остальном всё аналогично.

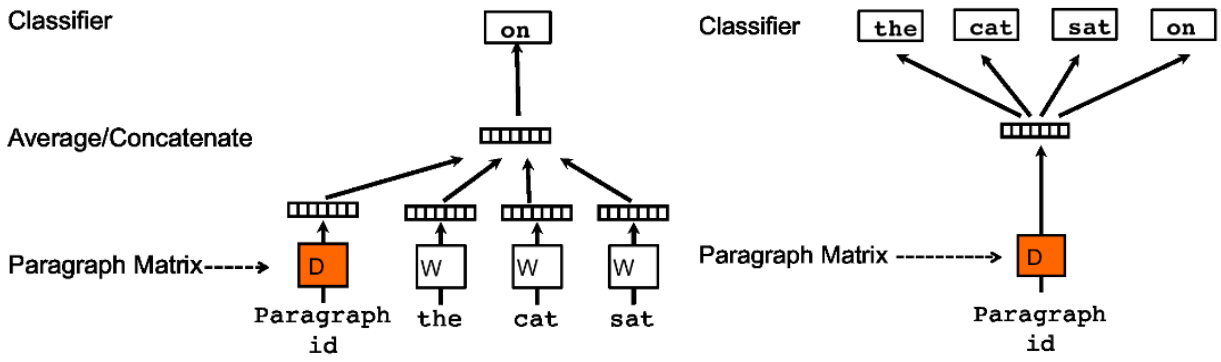



**Рисунок 1.2 - Разновидности doc2vec. DM и DBOW**

В статье [5] изложены рекомендации по гипер-параметрам для модели doc2vec, полученные эмпирическим путём. Также там указывается, что версия DBOW чаще работает лучше чем DM и что использование предобученных эмбеддингов слов улучшают качество эмбеддингов документов.

Bidirectional Encoder Representations from Transformers (BERT) [6] основывается на модели трансформер. Трансформер - модель, которая является альтернативой сверточным и рекуррентным моделям и использует механизм внимания (attention) для обработки последовательностей. BERT генерирует контекстуальные эмбеддинги для слов. То есть эмбеддинг слова зависит от его расположения в тексте и соседних слов, в отличии от модели word2vec, в которой эмбеддинг слова всегда одинаковый вне зависимости от контекста. BERT обучается на задаче предсказания пропущенных слов в предложении. Так как модель генерирует эмбеддинги слов, чтобы получить эмбеддинг предложения или документа, берется максимум или среднее по эмбеддингам всех слов. Sentence-BERT (SBERT) [7] - модификация, которая направлена на то, чтобы улучшить качество эмбеддингов предложений. На самом деле структура самой модели полностью совпадает с классическим BERT, но отличается процесс обучения. Она обучается на задаче предсказании сходства двух предложений.

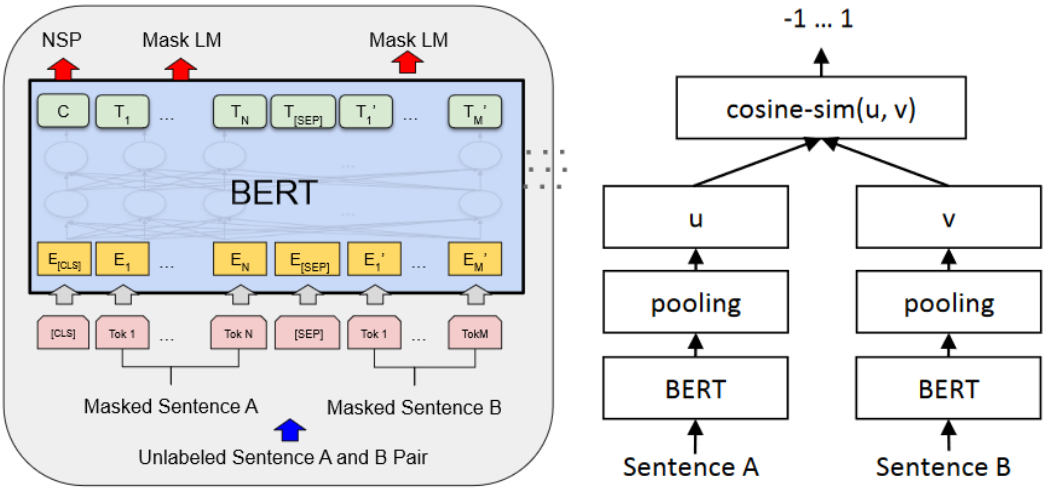



**Рисунок 1.3 - Модели BERT и SBERT**

Существуют и другие алгоритмы построения эмбеддингов для документов. Алгоритмы, описанные выше (кроме SBERT), не используют никакой разметки корпуса, однако есть алгоритмы которые опираются на нее, то есть используют обучение с учителем.

## 1.3 Постобработка эмбеддингов

### 1.3.1 Снижение размерности

Одна и проблем в машинном обучении это большие размерности в данных. Как следствие данные занимают много места, и, что еще важнее, некоторые алгоритмы долго работают на таких данных. Снижение размерности это уменьшение количества признаков. Конечно при снижении размерности теряется некоторая информация. Но зато алгоритмы быстрее обучаются на таких данных. Иногда они могут давать результаты чуть хуже, а иногда наоборот, так как уменьшение размерности может помочь убрать шумы в данных. Таким образом, снижение размерности используется для: сжатия данных, снижения шумов, а также для визуализаций (например снижение размеронсти до двух или трех, чтобы отобразить данные на графике). Методы в этой задаче можно разделить на линейные и нелинейные.

Метод главных компонент (principal component analysis, PCA) один из самых популярных методов. Он выполняет линейное отображение данных в пространство меньшей размерности так, чтобы сохранить максимально

возможную вариацию.  Использует ковариационную матрицу, для которой ищутся собственные вектора (главные компоненты) и собственные значения.

Uniform Manifold Approximation and Projection (UMAP) [8] - относительно новый алгоритм. Он является нелинейным. При работе UMAP строит взвешенный граф ближайших соседей, потом создает новый граф в низкоразмерном пространстве и приближает множество его ребер к исходному графу градиентным спуском. Метрика является параметром. Также важное свойство алгоритма это то, что обучившись он может повторять трансформацию на новых данных. Если сравнивать с другим нелинейным алгоритмом - t-SNE, то UMAP быстрее и  лучше обоснован математически, хотя часто визуально их результаты похожи.

### 1.3.2 Постобработка эмбеддингов слов

В статье [9] было замечено, что часто эмбеддингов слов имеют не нулевое стреднее и что большая часть вариации эмбеддингов содержится в небольшом подмножестве признаков. Также заметили, что главные компоненты эмбеддингов имеют зависимость с частотой встречаемости слова, что конечно нежелательно. На изображении по осям две главные компоненты, цвет - частота встречаемости:

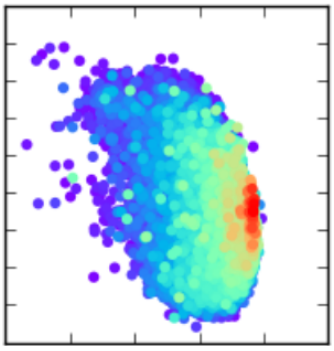


**Рисунок 1.4 - Главные компоненты эмбеддингов слов и частота**

Был предложен простой алгоритм постобработки:

1. Вычетается среднее из эмбеддингов.
2. Вычисляются k главных компонент PCA.
3. Из эмбеддингов удаляется информация об этих компонентах.

Было показано, что иногда данная постобработка улучшает результаты алгоритмов, которые используют эмбеддинги на входе.

## 1.4 Кластеризация

В данной работе используется кластеризация на основе плотности. Это вид кластеризации, в котором кластера определяются как области с более высокой плотностью, чем в остальном датасете. При таком подходе изолированные точки обычно помечаются как шум. Наиболее популярный алгоритм данного типа - основанная на плотности пространственная кластеризация для приложений с шумами (Density-based spatial clustering of applications with noise, DBSCAN). Он работает следующим образом: находятся точки, у которых соседей в окрестности больше заданного порога (core points), далее находятся связанные компоненты из таких точек, и потом оставшиеся точки (border points) относятся к установленным компонентам. Точки, которые не имеют соседей в окрестности, помечаются как шум (noise points).

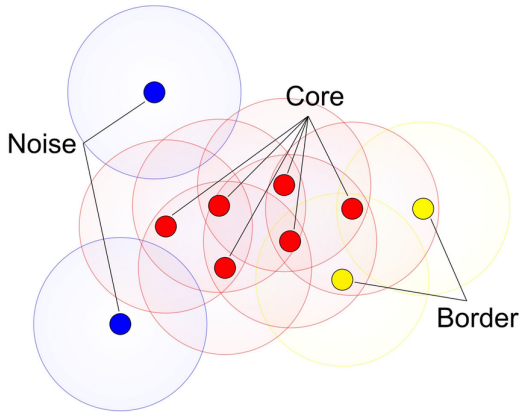



**Рисунок 1.5 - Типы точек в DBSCAN**

Худшая сложность алгоритма $O(n^2)$. Особенность этого метода это то, что кластеры могут получаться произвольной формы, в отличии, например, от алгоритма k-Means. Также количество кластеров определяется автоматически, а параметрами являются: метрика, размер окрестности и минимальное количество соседей в окрестности. HDBSCAN - модификация DBSCAN, которая опирается также на иерархичность в данных.

Также в данной работе использовалась иерархическая агломеративная кластеризация. Иерархическая кластеризация во время применения строит иерархию кластеров. Существует две стратегии построения такой иерархии: агломеративная и дивизионная. Агломеративная кластеризация инициализируется одноэлементными кластерами для каждого элемента

исходного набора данных. Далее постепенно “склеивает” эти кластера, пока не получится один кластер содержащий весь набор данных. На каждом шаге выбираются и объединяются ровно два кластера по определенному критерию. Дивизионная кластеризации работает наоборот. Она начинается с одного кластера, а далее рекурсивно разбивается на пары кластеров. Оба варианта алгоритма действуют жадно, то есть на каждом шаге выбирается оптимальный вариант склеивания (разбиения). В результате работы алгоритма получается дендрограмма, которая и описывает иерархию кластеров и данных.

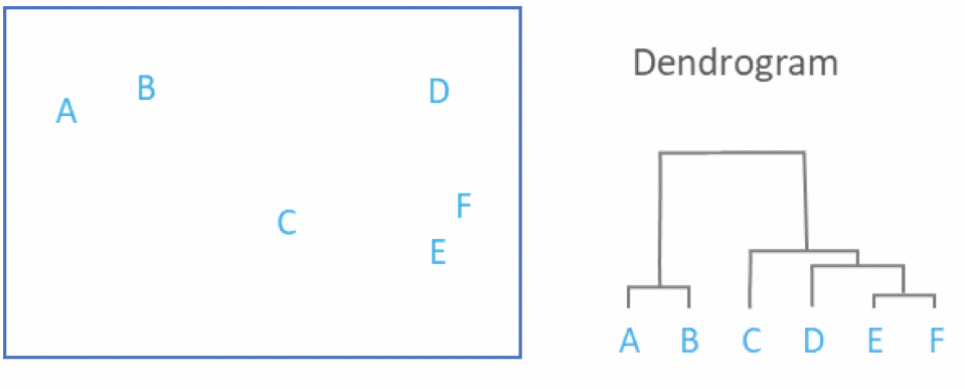



**Рисунок 1.6 - Пример дендрограммы**

Критерием объединения выступает расстояние между кластерами, то есть на каждом шаге объединяются самые близкие кластера. Одни из самых популярных метрик это подсчет минимального (метод одиночной связи), максимального (метод полной связи) или среднего (метод средней связи) расстояния из всех попарных расстояний между элементами двух кластеров. Худшая временная сложность алгоритма $O(n^3)$, но в некоторых случаях он оптимизирован до $O(n^2)$.

Методы проверки результатов кластеризации можно разделить на два типа: внешняя (external) и внутренняя (internal). Во внешней проверке используется известная классификация объектов, которая не была использована алгоритмом, а внутренняя использует только результаты самого алгоритма. Для внешней проверки могут использоваться методы проверки обычной классификации. В этом случае объектами являются пары точек, они могут принадлежать либо одному классу, либо разным. Если точки принадлежат одному классу и в результате кластеризации, и в размеченных

данных, то это считается true positive. Аналогично, также как в классификации определяются true negative, false positive, false negative. Используя эти значения можно посчитать метрики качества кластеризации:

$$rand\ index = \frac{TP+TN}{TP+TN+FP+FN}$$

$$precision = \frac{TP}{TP+FP}$$

$$recall = \frac{TP}{TP+FN}$$

# Глава 2. ПАЙПЛАЙН ОБРАБОТКИ

## 2.1 Данные

Основным источником данных в данной работе являются новостные статьи tut.by. Был написан парсер сайта, который извлекает текст статьи, заголовок, дату и теги для каждой новостной страницы. Количество собранных статей ~126000. Также использовались новостные сайты нескольких больниц.

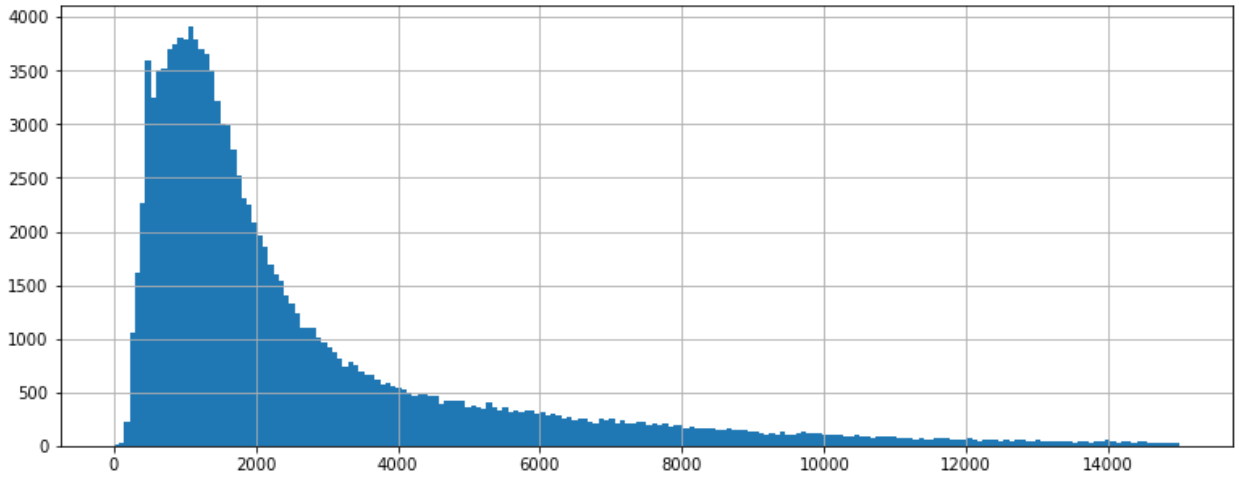



**Рисунок 2.1 - Гистограмма длин необработанных статей в символах**

К текстам применялась следующая предобработка:

1. Приведение к нижнему регистру
2. Удаление цифр и знаков пунктуации
3. Токенизация
4. Лемматизация
5. Удаление стоп-слов

Токенизация и лемматизация выполнялись одним пакетом - pymystem3. Этот пакет использует внешний исполняемый .exe файл при работе, он запускает его при обработке каждого документа. Из-за этого большое количество текстов обрабатывается очень долго. Для решения этой проблемы отдельные документы соединялись между собой специальным длинным разделителем, обрабатывались, и затем разбивались по разделителю обратно. Это даёт сильное ускорение, так как производилось гораздо меньше запусков .exe файла, а внутри он работает с достаточно хорошей скоростью.

## 2.2 Top2vec

Top2vec - алгоритм тематического моделирования (topic modeling) предложенный в статье [10]. Работает лучше чем классические алгоритмы для данной задачи: Latent Dirichlet Allocation (LDA) и Probabilistic Latent Semantic Analysis (PLSA). Top2vec менее нуждается в предобработке текста и сам выводит количество тем. Модель использует следующее свойство модели doc2vec: при построении эмбеддингов документов она также строит эмбеддинги слов, которые расположены в том же пространстве, что и эмбеддинги документов. То есть можно легко считать расстояние между эмбеддингами документов и слов. Алгоритм имеет следующие шаги:

1. Построение эмбеддингов Doc2vec.
2. Снижение размерности эмбеддингов алгоритмом UMAP.
3. Кластеризация эмбеддингов алгоритмом HDBSCAN.
4. Нахождение эмбеддингов тем (кластеров) как среднее эмбеддингов документов в кластерах.
5. Нахождение ключевых слов темы как ближайшие слова к эмбеддингу темы.

Таким образом, на выходе алгоритм даёт эмбеддинги документов, слов и тем, разбиение на темы и для каждой темы ключевые слова.

Применив top2vec к описанным ранее данным, были получены изображенные результаты. Каждая точка - эмбеддинг документа (размерность снижена до двух с помощью UMAP). На изображении точки раскрашены в соответствии с тегами, которые были взяты при парсинге статей. Видно, что документы с одинаковым тегом находятся близко.

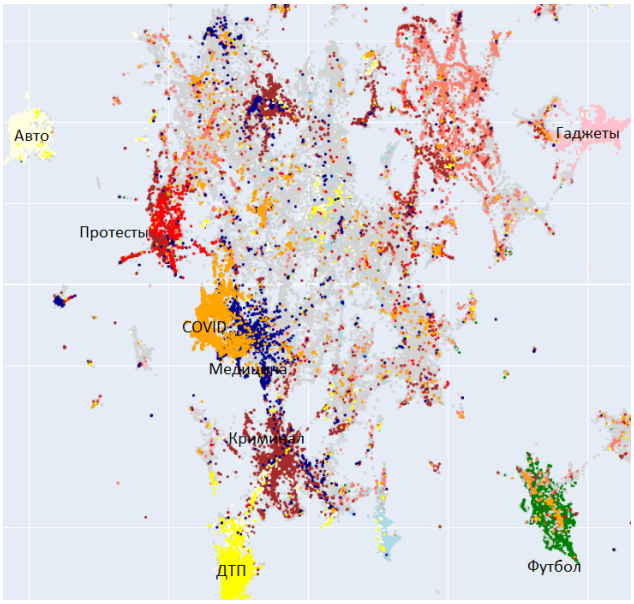



**Рисунок 2.2 - Эмбеддинги документов после UMAP**

Для следующего изображения были взяты тег “гаджеты” и кластер, который имеет наибольшее пересечение с этим тегом по документам. Документы которые попадают в пересечение - зелёные, документы которые имеют тег, но не попали в кластер - оранжевые. Этот пример показывает, что кластер является подмножеством документов с заданным тегом.

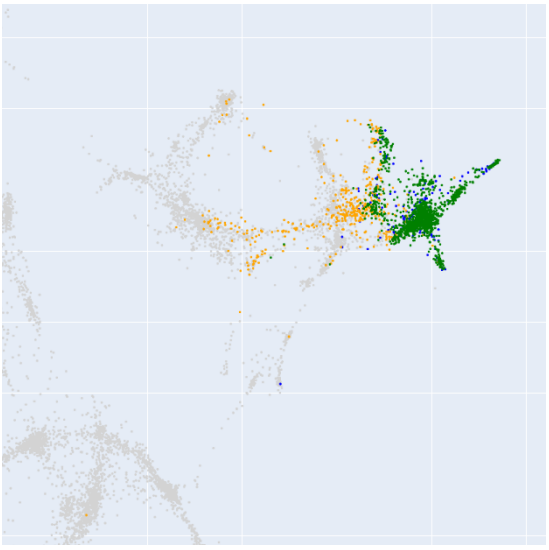


**Рисунок 2.3 - Эмбеддинги с тегом “гаджеты” и кластер**

Следующая таблица показывает связь тегов и кластеров, которые имеют наибольшие пересечения. Также отображены ключевые слова в кластерах.

| | topic | tag | n_topic | n_tag | intersection | iou | part_topic | part_tag | topic_words |
|---|---|---|---|---|---|---|---|---|---|
| **1042** | 0 | ДТП | 4356 | 4940 | 4061 | 0.775740 | 0.932277 | 0.822065 | [дтп, пдд, водитель, гаи, встречок] |
| **900** | 1 | футбол | 2538 | 5028 | 2452 | 0.479468 | 0.966115 | 0.487669 | [бате, борисовчанин, городея, еврокубок, еврок... |
| **2847** | 2 | Хоккей | 2257 | 2952 | 2234 | 0.750924 | 0.989809 | 0.756775 | [хоккеист, шаиба, кхл, буллит, овертаим] |
| **1721** | 15 | Погода | 1520 | 4231 | 1480 | 0.346523 | 0.973684 | 0.349799 | [погода, дождь, ветер, порыв, похолодать] |
| **2351** | 6 | вооружения | 1828 | 3674 | 1457 | 0.360198 | 0.797046 | 0.396570 | [бомбардировщик, многоцелевои, надводныи, свер... |
| **978** | 19 | футбол | 1478 | 5028 | 1385 | 0.270455 | 0.937077 | 0.275457 | [атлетико, ливерпуль, реал, мадридскии, ювентус] |
| **2927** | 18 | Гаджеты | 1486 | 2880 | 1334 | 0.439974 | 0.897712 | 0.463194 | [полноэкранныи, fonearena, nubia, цветопередач... |
| **1558** | 31 | Автомобили | 1347 | 4390 | 1323 | 0.299728 | 0.982183 | 0.301367 | [привод, трансмиссия, акп, переднии, бензиновыи] |
| **4727** | 18 | смартфоны | 1486 | 1817 | 1290 | 0.640835 | 0.868102 | 0.709961 | [полноэкранныи, fonearena, nubia, цветопередач... |
| **2097** | 12 | Коронавирус | 1589 | 3953 | 1267 | 0.296374 | 0.797357 | 0.320516 | [заражение, инфицирование, инфицированныи, cov... |

**Таблица 2.1 - Отношение тем и тегов**

Где n_topic - количество документов в кластере
n_tag - количество документов с заданным тегом
iou - intersection over union

$$part\_topic = \frac{intersection}{n\_topic}$$

$$part\_tag = \frac{intersection}{n\_tag}$$

График отображающий долю статей от их общего количества в неделю по выбранным темам (кластерам):

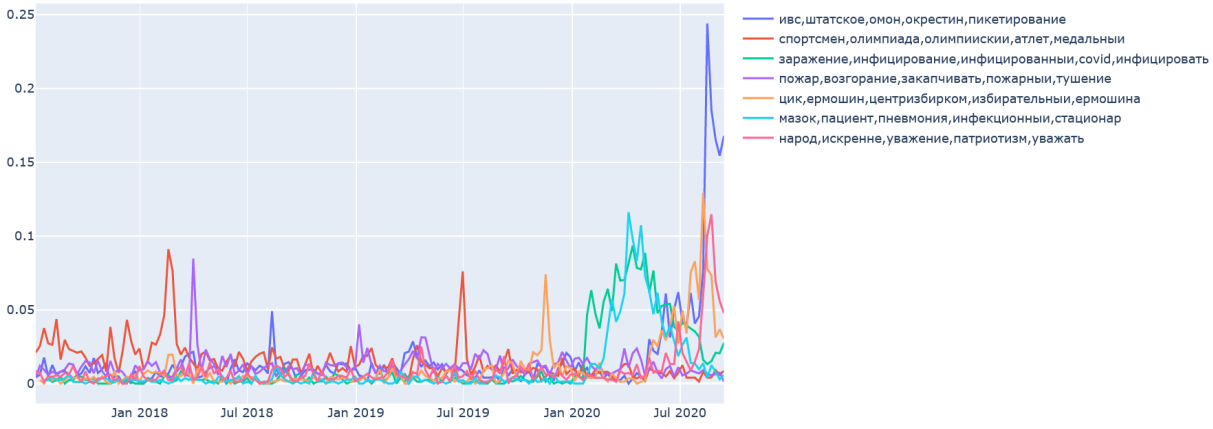



**Рисунок 2.4 - Тренды тем**

Таким образом можно заключить, что эмбеддинги в данном методе хорошо отражают семантические свойства данных.

## 2.3 Обобщенный пайплайн

В данной работе предлагается обобщение top2vec, путем изменения некоторых шагов и добавления новых. В том числе проверок, которые могут помочь выбрать лучшую комбинацию алгоритмов для конкретных данных.

Этапы:

1. Предобработка текста. Несмотря на то, что top2vec может работать и без неё, было замечено, что это может улучшить результаты.
2. Построение эмбеддингов документов.
3. Постобработка эмбеддингов.
4. Оценка эмбеддингов.
5. Кластеризация, тематическое моделирование.
6. Оценка кластеризации.
7. Нахождение ключевых слов в кластерах.

## 2.4 Построение эмбеддингов документов

На данном этапе top2vec использует doc2vec. В экспериментах этой работы doc2vec обучался на очищенных текстах в отличии от top2vec. Использовалась реализация в пакете gensim со следующими параметрами: 'dm': 0, 'vector_size': 300, 'window': 15, 'min_count': 50, 'sample': 1e-5, 'epochs': 40, 'hs': 1, 'negative': 0, 'dbow_words': 1. Обучение длилось 2.5 часа.

Помимо doc2vec для сравнения использовались и другие алгоритмы:

Взвешенная сумма эмбеддингов word2vec с весами idf. Также обучалась на очищенных текстах. Использовалась реализация в пакете gensim со следующими параметрами: 'size': 300, 'window': 5, 'min_count': 20, 'sg': 1, 'negative': 5, 'sample': 1e-5, 'iter': 150. Обучение длилось 4 часа. Выборочно были проверены семантические свойства полученных векторов слов. Например, поиск слов со схожим значением (числа на изображении - значения косинуса между исходным словом и схожим).

```
Request: вич
0.764 - вичинфекция
0.620 - спид
0.619 - гепатит
0.613 - антиретровирусный
0.593 - вичположительный
0.590 - вичинфицированный
0.571 - иммунодефицит
```

**Рисунок 2.5 - Пример поиска по эмбеддингам word2vec**

Модели BERT и SBERT. Так как обучение этих моделей очень времязатратно, использовались предобученные на русском языке модели из пакета Deeppavlov. Время получения эмбеддингов - 5.5 часов. В данном случае предобработка входных текстов не использовалась, так как это ухудшает работу BERT. У реализации в пакете Deeppavlov есть ограничение на длину входных текстов для модели, поэтому подавались не целые документы, а первые 1000 символов.

Также пробовалась взвешенная сумма эмбеддингов BERT с весами idf. Но этот вариант очень затратен по памяти и по времени. А результаты были сильно хуже других методов, поэтому дальше этот метод не рассматривается.

## 2.5 Постобработка эмбеддингов

В пункте 1.3.2 описана постобработка эмбеддингов слов. В данной работе было замечено, что эмбеддинги документов также могут обладать свойствами, которыми обладают эмбеддингов слов. Например, для эмбеддингов BERT:

- Норма среднего вектора $norm(emb.mean) = 6.92$, а средняя норма по всем векторам $norm(emb).mean = 8.65$. После применения алгоритма постобработки имеем: $norm(processed.mean) = 0$, $norm(processed).mean = 4.69$.
- Большая часть дисперсии содержится в первых компонентах. На графике по горизонтальной оси - номер компоненты PCA в логарифмической шкале, по вертикальной - доля дисперсии для компонент.

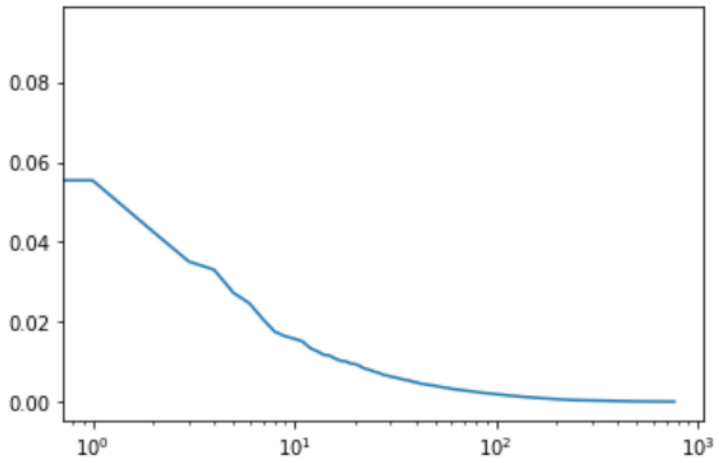



**Рисунок 2.6 - Дисперсии компонент PCA**

- Первые компоненты могут содержать информацию о длине документа. На графике по осям координаты эмбеддингов в двух главных компонентах. Жёлтым цветом помечены эмбеддинги, которые соответствуют документам с длиной больше 10000.

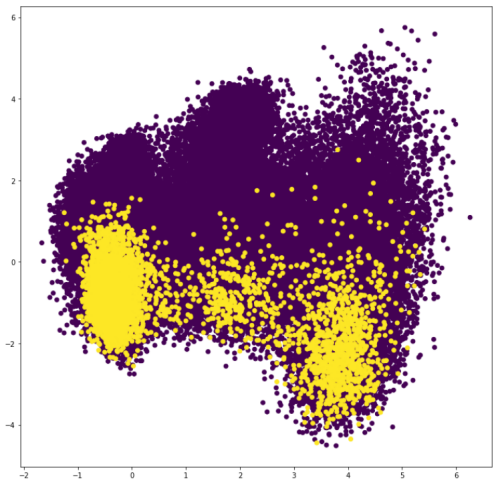


**Рисунок 2.7 - Главные компоненты эмбеддингов и длина**

Таким образом алгоритм постобработки также можно пробовать применять к эмбеддингам документов.

В top2vec постобработкой можно назвать уменьшение размерности алгоритмом UMAP. Это необходимый шаг, так как он помогает бороться с проблемами метрик в высокой размерности и ускоряет дальнейшие шаги.

Алгоритм UMAP запускался со следующими параметрами: n_neighbors=15, n_components=5, metric='cosine'.

## 2.6 Оценка эмбеддингов

Оценка эмбеддингов нужна для сравнения различных комбинаций алгоритмов построения эмбеддингов, предобработки теста, постобработки эмбеддингов и для выбора лучшего варианта.

Один из способов применения эмбеддингов является поиск схожих документов. В имеющихся данных для каждого документа есть набор тегов, полученный из источника данных. Будем считать документы похожими, если у них есть общий тег. Тогда в качестве оценки качества эмбеддингов возьмём величину равную вероятности того, что документы имеют одинаковый тег, при условии, что косинусное расстояние (единица минус косинус) между эмбеддингами меньше заданного. То есть $P(same\ tag \mid distance\ <\ x)$. Чем больше эта величина для заданного x, тем лучше. Чтобы посчитать эту величину для всевозможных x, применим теорему Байеса:

$$P(A|B)\ =\ \frac{P(B|A)*P(A)}{P(B)},$$

где A - “документы имеют общий тег”, B - “расстояние меньше заданного x”.

Тогда имеем:

$$P(same\ tag|distance\ <\ x)\ =\ \frac{P(distance<x|same\ tag)*\ P(same\ tag)}{P(distance<x)}\ =\ \frac{F_{same\ tag}(x)*P(same\ tag)}{F(x)}$$

где F(x) - функция распределения расстояний.

Аналогично можно посчитать:

$$P(same\ tag|distance\ >\ x)\ =\ \frac{P(distance>x|same\ tag)*P(same\ tag)}{P(distance>x)}$$

$$=\ \frac{(1-P(distance<x|same\ tag))*P(same\ tag)}{1-P(distance<x)}\ =\ \frac{(1-F_{same\ tag}(x))\ *\ P(same\ tag)}{1-F(x)}$$

В качестве функции распределения расстояний бралась выборочная функция распределения. На изображении слева синим и оранжевым соответственно примеры графиков функций плотности вероятности $P(distance)$ и $P_{same\ tag}(distance)$ для эбмеддингов doc2vec (в приложениях также представлены для эбмеддингов word2vec и BERT). Спарва функция $P(same\ tag|distance\ <\ x)$.

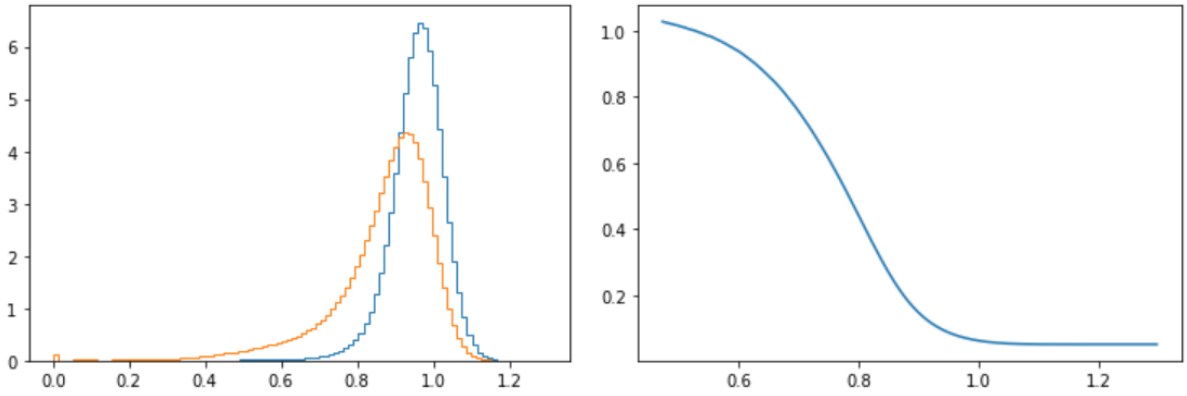



**Рисунок 2.8 -** $P(distance)$ и $P_{same\ tag}(distance)$**.** $P(same\ tag|distance < x)$

Как значение оценки эмбеддинга бралось значение функции $P(same\ tag|distance < x)$ в точке $x = quantile_{P(distance)}(0.01)$. Результаты оценок для различных эмбеддигов (PP - алгоритм из пункта 1.3.2):

| [P] | Без обработки | PP | UMAP | PP+UMAP |
|---|---|---|---|---|
| doc2vec | 0.670 | 0.647 | 0.679 | 0.624 |
| BERT | 0.453 | 0.599 | 0.432 | 0.522 |
| word2vec+idf | 0.588 | 0.690 | 0.574 | 0.596 |

**Таблица 2.2 - Оценки эмбеддигов**

Таким образом, по данной метрике лучше всего себя показывает word2vec+idf с постобработкой. Это значит, что для поиска похожих документов по расстоянию между эмбеддингами, лучше использовать этот алгоритм.

Альтернативным вариантом сравнения эмбеддингов было обучение однослойной нейронной сети (или softmax регрессии), которая на вход получает эмбеддинг, а на выходе предсказывает тег. Эмбеддинги, которые дают лучший результат на этой модели, и будут считаться лучшими. Но этот вариант менее стабильный и в нём надо подбирать параметры для обучения модели отдельно для каждых эмбеддингов. Поэтому предпочтению отдано предыдущему вариану оценки.

Также можно осуществлять выборочную проверку путем поиска самых похожих документов для заданного. Пример для модели doc2vec с косинусной метрикой:

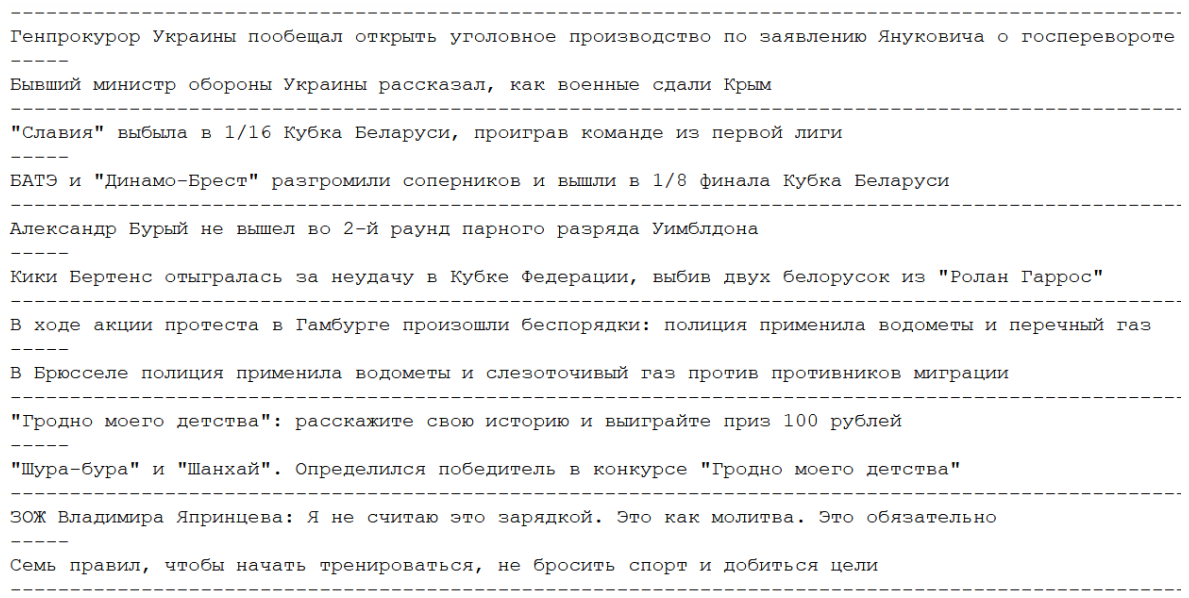



**Рисунок 2.9 - Примеры похожих документов по эмбеддингам doc2vec**

## 2.7 Кластеризация

Top2vec использует HDBSCAN для кластеризации. Этот алгоритм достаточно быстрый, легко настраивается и хорошо работает после UMAP, так как использует локальные свойства точек. На вход подавались только эмбеддинги после UMAP, так как без уменьшения размерности алгоритм будет работать существенно дольше. Использовалась реализация из python пакета hdbscan со следующими параметрами: min_cluster_size=15, metric='euclidean', cluster_selection_method='eom'.

Оценка результатов кластеризации значениями precision и recall (пункт 1.4) в зависимости от входных эмбеддингов:

| [precision - recall] | UMAP | PP+UMAP |
|---|---|---|
| doc2vec | 0.747 - 0.026 | 0.692 - 0.019 |
| BERT | 0.699 - 0.017 | 0.680 - 0.027 |
| word2vec+idf | 0.792 - 0.032 | 0.766 - 0.030 |

**Таблица 2.3 - Оценки после кластеризации**

Значения recall такие небольшие, так как кластера получаются сильно меньше, чем подмножества документов с один тегом. Это можно исправить либо изменением параметров кластеризации, либо последовательным слиянием близких кластеров, как это делается в top2vec. Значение rand index не приводится, так как оно везде близко к единице и не информативно.

По данным измерениям лучшие результаты кластеризации дают эмбеддинги word2vec+idf.

## 2.8 Ключевые слова

Чтобы отобразить семантическое значение каждой темы, вычисляются ключевые слова для каждого кластера. Это осуществляется следующим образом. Текстовые документы группируются по темам и объединяются. Далее считаются значения TF (term frequency) для получившихся объединенных документов. Отдельно считаются значения IDF (inverse document frequency) для всех слов словаря по изначальным необъединенным документам. Потом считаются значения TF-IDF для объединенных документов по формуле $tf * idf^2$. И для каждого такого документа выбирается заданное число слов (использовалось 5 слов) с наибольшим значением TF-IDF. Такие образом, получаем ключевые слова для каждой темы. Псевдокод описанного процесса:

```
documents = array of pairs (document, topic_id)
topic_documents = documents.groupBy(topic_id).concat(document)
tf = tf(topic_documents)
idf  = idf(documents)
tf_idf = tf * idf^2
topic_words = tf_idf.argsort(desc)[:, :n_topic_words]

Shapes:
documents: (n_documents, 2)
topic_documents: (n_topics, 2)
tf: (n_topics, n_words)
idf: (n_words)
tf_idf: (n_topics, n_words)
topic_words: (n_topics, n_topic_words)
```

Примеры полученных ключевых слов можно увидеть в пункте 2.9.

## 2.9 Структура кластеров

Для лучшего представления взаимоотношений получившихся кластеров тем после алгоритма HDBSCAN также применяется агломеративная кластеризация. Для этого считается средний вектор эмбеддинг среди всех эмбеддингов для каждой темы и агломеративная кластеризация применяется к получившейся выборке с количеством элементов равным количеству тем. Как параметры агломеративной кластеризации используются косинусная метрика и метод полной связи. Результатом кластеризации является дендрограмма, которая отображает структуру изначальных кластеров. Пример фрагмента такой дендрограммы (как метки кластеров используются ключевые слова тем из пункта 2.8):


**Рисунок 2.10 - Фрагмент дендрограммы**

## 2.10 Итоговый пайплайн

Таким образом, после сравнения различных алгортмов построения эмбеддингов документов и их обработки, приходим к следующема пайплайну тематического моделирования:

1. Предобработка текста из пункта 2.1.
2. Построение эмбеддингов документов word2vec+idf.
3. Постобработка эмбеддингов алгоритмом снижения размерности UMAP.
4. Кластеризация HDBSCAN.
5. Нахождение ключевых слов в кластерах с помощью модифицированного алгоритма TF-IDF из пункта 2.8.
6. Построение дендрограммы кластеров.

Как было показано в пункте 2.6, к шагу 3 может быть добавлен алгоритм из пункта 1.3.2. Это может улучшить качество поиска схожих документов по расстоянию между эмбеддингами, но для последующей кластеризации он излишен.

Основными отличиями от пайплайна top2vec являются:

- Добавление предобработки текста.
- Изменение алгоритма построения эмбеддингов с doc2vec на word2vec+idf, что обосновано в пункте 2.7.
- Другой способ нахождения ключевых слов в кластерах.

# Глава 3. ОБЗОР ТЕХНОЛОГИЙ

## 3.1 Хранение данных

Распространенным способом хранения данных являются базы данных, но они относительно ограничены в объеме хранящихся данных. Поэтому для хранения больших данных часто используют распределенную файловую систему, например HDFS (Hadoop Distributed File System). HDFS быстрая, так как работает на кластере. Каждый файл разделяется на блоки размером 128 МБ, которые распределяются по кластеру. Каждый блок может быть продублирован несколько раз (по умолчанию три) на разных узлах кластера, чтобы повысить надёжность системы. Машины в кластере делятся на два типа: NameNode и DataNode. Первый отслеживает метаданные блоков, а второй хранит сами данные. Одно из важнейших свойств HDFS - это масштабируемость, то есть при необходимости можно добавлять узлы в использующийся кластер, тем самым увеличивая максимальный объем хранилища.

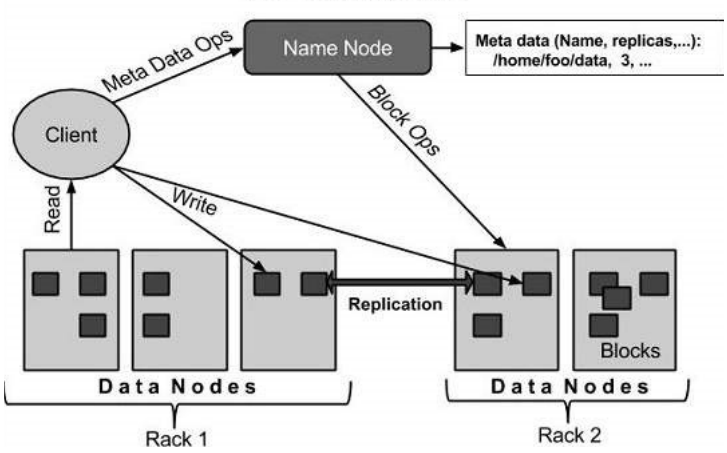



**Рисунок 3.1 - Архитектура HDFS**

Delta Lake - формат таблиц, хранящихся на (распределенной) файловой системе. Файлы в delta таблице имеют бинарный формат parquet, который использует сжатие и содержит схему данных. Для хранящихся таблиц может быть применено партиционирование данных. Например, если в таблице есть колонка “год”, записи могут быть разбиты на группы (партиции) по значению этой колонки, и каждая группа сохранится в отдельную папку. Тогда, чтобы

прочитать данные за конкретный код, не надо читать всю таблицу, а только нужные файлы. Также поддерживаются операции upsert и delete аналогичные операциям баз данных. Delta поддерживает версионность данных и хранит всю историю транзакций над таблицей. В любой момент можно откатить последние изменения таблицы до определённой версии. Delta полность совместима с Spark, а это значит что и чтение, и запись таблиц происходит параллельно.

## 3.2 Распределённые вычисления

Одной из самых популярных библиотек для обработки данных на кластере является Apache Spark. Spark может выполнять задачи обработки очень больших объемов данных.

Spark приложение состоит из двух частей: драйвера, который преобразует код пользователя в задачи, которые могут быть распределены между рабочими узлами, и исполнителей, которые запускаются на этих узлах и выполняют назначенные им задачи.

При обработке пользовательского кода строится направленный ациклический граф (DAG) задач. Он определяет, какие задачи выполняются на каких узлах и в какой последовательности. Также одна из основных концепций Spark это Resilient Distributed Dataset (RDD) - абстракция представляющая неизменяемую коллекцию объектов, распределенных на кластере. Операции над такой коллекцией параллельны. RDD разбивается на партиции, которые и распределены на кластере. Все операции над данными выполняются в памяти, объекты также хранятся в памяти, но при её нехватке могут быть кэшированы на диске.

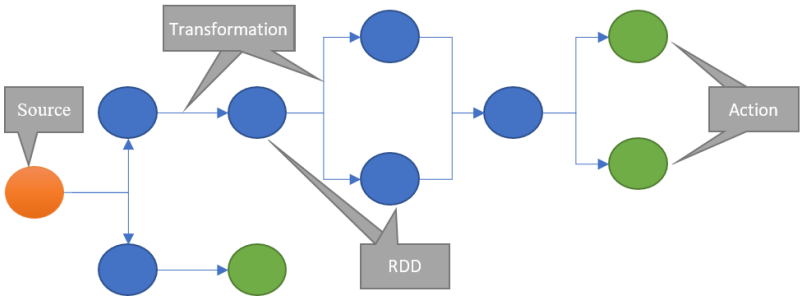



**Рисунок 3.2 - Пример DAG**

В Spark есть два типа операций: трансформации и действия (action). Трансформации применяются для создания новых RDD, а действия это операции при которых данные будут переданы на драйвер программы или записаны на диск. Пока не вызвано действие, Spark не выполняет трансформации, а просто хранит их как DAG не вычисляя.

## 3.3 Оркестрация

Оркестратор пайплайнов это инструмент, который помогает собрать все процессы в одном месте и автоматизировать их выполнение. Обычно оркестратор имеет функционал, который может запускать нужные пайплайны по рассписанию, например, с каким-либо интервалом времени. Также ведётся учёт всех запущенных пайплайнов и их состояний. Любой оркестратор должен предоставлять способ определения пайплайна. Это может выполняться через программный (API) или визуальным интерфейс. Как правило пайплайн в оркестраторе представляется в виде направленного ациклического графа, в котором вершины это процессы, а дуги это зависимости и отношения между ними. Если какой то процесс завершился ошибкой, то следующий за ним в таком графе не будет запущен. Так как оркестратор это отдельное приложение, он требует отдельной установки и поддержки. Популярными оркестраторами являются: Apache Airflow, Dagster, Prefect.

## 3.4 Облачные вычисления

Облако - это сервера доступные через интернет с установленными программным обеспечением и базами данных, которые работают на этих серверах. Использование облака освобождает компании от управления физическими ресурсами, а также поддержки ПО на этих ресурсах.

Одной из главных технологий, которая позволила использовать облачные вычисления, является виртуализация. Виртуализация позволяет создать виртуальную машину, которая ведет себя так, как если бы это был физический компьютер. Виртуальные машины на одной и той же физической машине изолированы друг от друга, то есть они не взаимодействуют друг с другом, а данные и приложения с одной виртуальной машины не видны другим виртуальным машинам.

Выделяют три вида моделей облачных вычислений: IaaS, PaaS, SaaS. Модель Infrastructure-as-a-Service предоставляет серверы и хранилища в

облаке. Platform-as-a-Service предоставляет инструменты разработки, операционные системы, инфраструктуру и так далее, то есть все необходимое для создания приложения. Software-as-a-Service предоставляет готовые к использованию приложения в облаке.

На текущий момент самыми популярными облачными решениями являются Amazon Web Services, Microsoft Azure и Google Cloud Platform.

# Глава 4. АРХИТЕКТУРА СИСТЕМЫ

В данной главе описана архитектура системы для сбора и агрегации статей и документов из различных постоянно обновляющихся источников, а также обработки полученных данных с применением пайплайна из пункта 2.10.

## 4.1 Данные и пайплайн

В системе используются данные со следующих источников: tut.by, naviny.online, komzdrav-minsk.gov.by, 4gkb.by.

Для того чтобы включить новый источник данных в систему, нужно чтобы источник предоставлял какой-либо способ итерирования по статьям (документам), тогда можно будет программно собирать статьи. Например, в случае источника tut.by состоянием итерации является число из ссылки на статью (в ссылке https://news.tut.by/721783.html состояние - 721783). При увеличении этого числа получится ссылка на следующую статью. Для источника naviny.online состояние это дата (ссылка https://naviny.online/day/2021/03/12, состояние 2021/03/12). При работе используется таблица в базе данных, которая содержит поля process_name, state_start и state_stop. В ней каждый процесс записывает состояние с которого он начал работать и на каком закончил. Таким образом, алгоритм проверки новых данных в источнике (результат работы в случае с веб сайтами - ссылки на новые необработанные статьи):

1. Для текущего источника получить последнее состояние, на котором была остановлена работа ранее.
2. Создать в таблице запись о начале работы со стартовым состоянием.
3. Инкрементально итерироваться по состояниям пока это возможно и генерировать ссылки на новые статьи.
4. После окончания работы обновить стартовую запись последним обработанным состоянием.

Далее ссылки записываются в таблицу и запускается следующий шаг пайплайна.

Похожая схема используется для всех остальных этапов пайплайна. При записи данных в любую таблицу добавляется колонка с временем последнего обновления. Это помогает следующему процессу пайплайна считывать только

новые данные, которые не были обработаны ранее. Используется та же таблица состояний, только как состояние принимается время обновления записи в таблице. Таким образом, реализована инкрементальная загрузка данных и концепция change data capture.

Пайплайн включает в себя этапы:

1. Генерация ссылок на новые статьи в источнике.
2. Скачивание html страниц по ссылкам.
3. Разбор html страниц и извлечение нужных данных.
4. Интеграция данных разных источников в одну таблицу.
5. Предобработка текста.
6. Построение эмбедингов документов моделью word2vec+idf.
7. Уменьшение размерности алгоритмом UMAP.
8. Кластеризация HDBSCAN.
9. Нахождение ключевых слов для кластеров.
10. Построение дендрограммы кластеров.
11. Преобразования полученных данных для дальнейшего использования (например для визуализаций).

Каждый этап реализован в виде модуля на языке python. Первые 6 шагов используют библиотеку Spark. Для оркестрации пайплайна использовался инструмент Dagster. Поэтому каждый python модуль обернут в Dagster блок, что позволяет конструировать пайплайн из этих блоков. Так выглядит часть описанного пайплайна (только для двух источников данных) в Dagster UI:

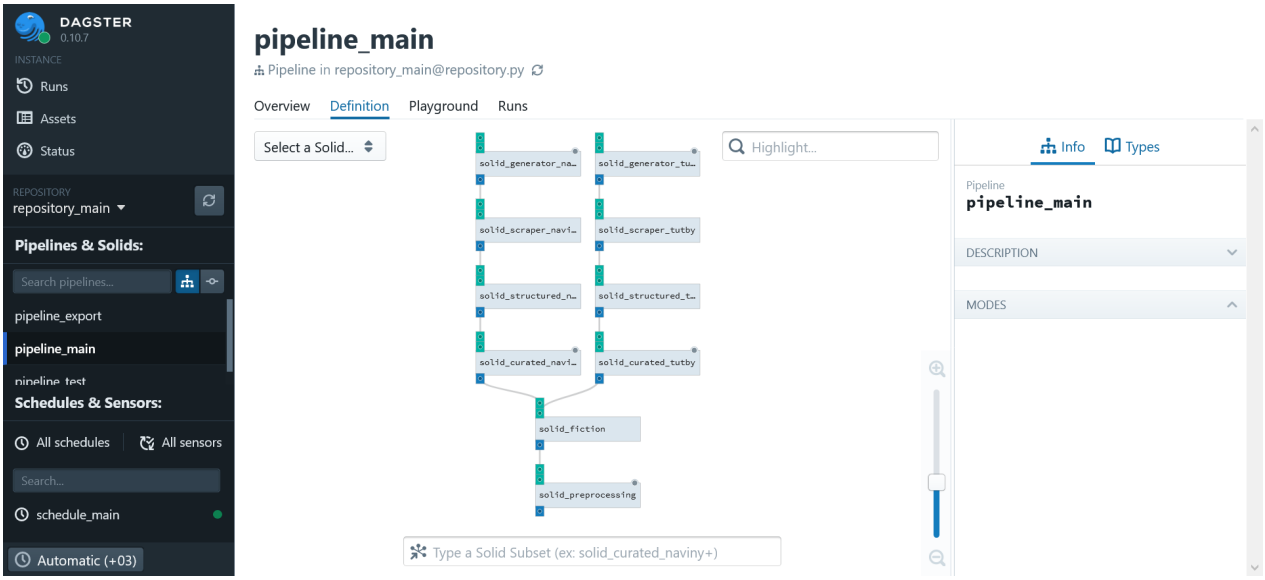



**Рисунок 4.1 - Dagster UI**

Сначала для каждого источника генерируются ссылки, скачиваются статьи и парсятся, потом они собираются в одну таблицу и на этой таблице запускается предобработка текстов.

## 4.2 Хранилище данных

```
datalake
- raw
  - url
    - {source}.delta
  - html
    - {source}.delta
- structured
  - {source}.delta
- curated
  - article.delta
- analytics
  - preprocessed.delta
  - embedding.delta
```

**Рисунок 4.2 - Структура хранилища данных**

Хранилище (или data lake) разделяется на 5 зон: сырые данные (raw), структурированные данные (structured), очищенные и интегрированные данные (curated), данные для аналитики (analytics), данные для внешних приложений

(consumer). В сырых данных хранятся ссылки и html страницы. В структурированных - данные, полученные после парсинга веб страниц. В зонах raw и structured данные разделены по источникам, то есть для каждого источника используется своя таблица. В curated хранится таблица article, которая содержит все статьи. В зоне аналитики находятся все данные полученные с помощью применения алгоритмов машинного обучения (например, после предобработки текстов, вычисления эмбеддингов и т.д.). Также есть зона consumer, в которую попадают данные приведенные в нужный формат для непосредственного использования каким либо приложением или дашбордом (например из пункта 4.4). Также предусмотрена зона для плохих данных (bad data), в которую могут записываться данные, которые по каким-либо причинам не перешли на следующий слой (например, нераспаршенная веб страница). Все таблицы хранятся в формате delta lake.

### 4.3 Развертывание системы

Для работы системы необходимо три компоненты: хранилище данных, Spark кластер для обработки и база данных для хранения таблицы состояний. Dagster запускается на главном узле кластера с расписанием запусков пайплайна каждые три часа. Для развертывания была выбрана облачная платформа Google Cloud Platform (GCP). Спроектированная система использует следующие сервисы GCP:

- Storage - хранилище данных.
- Dataproc - Spark кластер.
- Postgresql - база данных.

### 4.4 Визуализация

Чтобы продемонстрировать результаты работы системы и пайплайна обработки данных, дополнительно был реализован сайт-дашборд. Сайт имеет 4 вкладки:

- На первой вкладке после выбора темы из списка отображаются график популярности во времени выбранной темы и последние статьи.
- На второй вкладке после выбора какой-либо статьи отображаются самые похожие статьи на выбранную. Расстояние считается между эмбеддингами как единица минус косинус.

- Третья вкладка отображает артикли в семантическом 2D пространстве. Для снижения размерности эмбеддингов используется UMAP.
- Четвертая содержит полную дендрограмму тем из пункта 2.9.

Для реализации использовалась библиотека streamlit. Также этот инструмент предоставляет возможность развертывания сайта. Ссылка на реализованный сайт: share.streamlit.io/sinytim/topics. В приложениях приведены скриншоты сайта.

# ЗАКЛЮЧЕНИЕ

Таким образом, был реализован пайплайн для построения эмбеддингов документов и проведения тематического моделирования на их основе. Были введены метрики качества на используемых данных и выбран лучший алгоритм в соответствии с ними. Также предложен алгоритм нахождения ключевых слов для тем.

Разработана система для сбора статей из различных постоянно обновляющихся источников и применения построенного пайплайна для их обработки. Дополнительно реализован сайт для визуализации результатов работы системы.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Efficient Estimation of Word Representations in Vector Space, 2013. [Электронный ресурс]. - Режим доступа: https://arxiv.org/pdf/1301.3781.pdf - Дата доступа: 01.05.2021
[2] Global Vectors for Word Representation, 2014. [Электронный ресурс]. - Режим доступа: https://nlp.stanford.edu/pubs/glove.pdf - Дата доступа: 01.05.2021
[3] Word Embeddings. [Электронный ресурс]. - Режим доступа: https://lena-voita.github.io/nlp_course/word_embeddings.html - Дата доступа: 01.05.2021
[4] Distributed Representations of Sentences and Documents, 2014. [Электронный ресурс]. - Режим доступа: https://arxiv.org/pdf/1405.4053.pdf - Дата доступа: 01.05.2021
[5] An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation, 2016. [Электронный ресурс]. - Режим доступа: https://arxiv.org/pdf/1607.05368v1.pdf - Дата доступа: 01.05.2021
[6] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019. [Электронный ресурс]. - Режим доступа: https://arxiv.org/pdf/1810.04805.pdf - Дата доступа: 01.05.2021
[7] Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, 2019. [Электронный ресурс]. - Режим доступа: https://arxiv.org/pdf/1908.10084.pdf - Дата доступа: 01.05.2021
[8] UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, 2020. [Электронный ресурс]. - Режим доступа: https://arxiv.org/pdf/1802.03426.pdf - Дата доступа: 01.05.2021
[9] All-but-the-top: simple and effective post-processing for word representations, 2018. [Электронный ресурс]. - Режим доступа: https://arxiv.org/pdf/1702.01417.pdf - Дата доступа: 01.05.2021
[10] Top2vec: distributed representations of topics, 2020. [Электронный ресурс]. - Режим доступа: https://arxiv.org/pdf/2008.09470.pdf - Дата доступа: 01.05.2021
[11] HDFS Architecture. [Электронный ресурс]. - Режим доступа: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.pdf - Дата доступа: 01.05.2021
[12] Delta Lake. [Электронный ресурс]. - Режим доступа: https://delta.io - Дата доступа: 01.05.2021

[13] Apache Spark. [Электронный ресурс]. - Режим доступа: https://www.ibm.com/cloud/learn/apache-spark - Дата доступа: 01.05.2021
[14] Dagster. [Электронный ресурс]. - Режим доступа: https://dagster.io - Дата доступа: 01.05.2021
[15] What is the cloud. [Электронный ресурс]. - Режим доступа: https://www.cloudflare.com/learning/cloud/what-is-the-cloud - Дата доступа: 01.05.2021

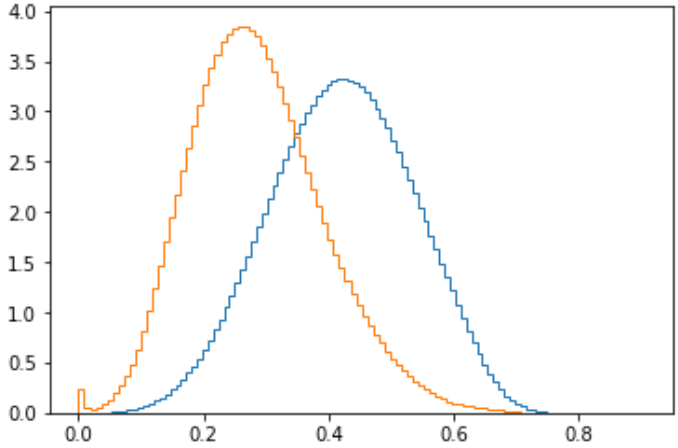
# ПРИЛОЖЕНИЯ

## Выборочные функций плотности вероятности

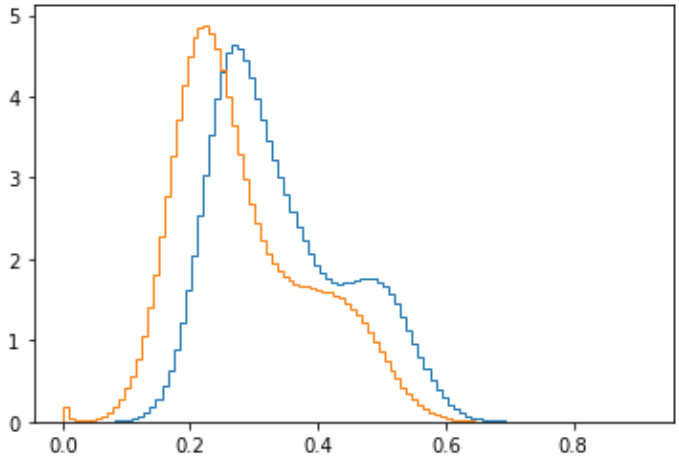
Синим - $P(distance)$

Оранжевым - $P_{same\ tag}(distance)$

Для эбмеддингов word2vec:

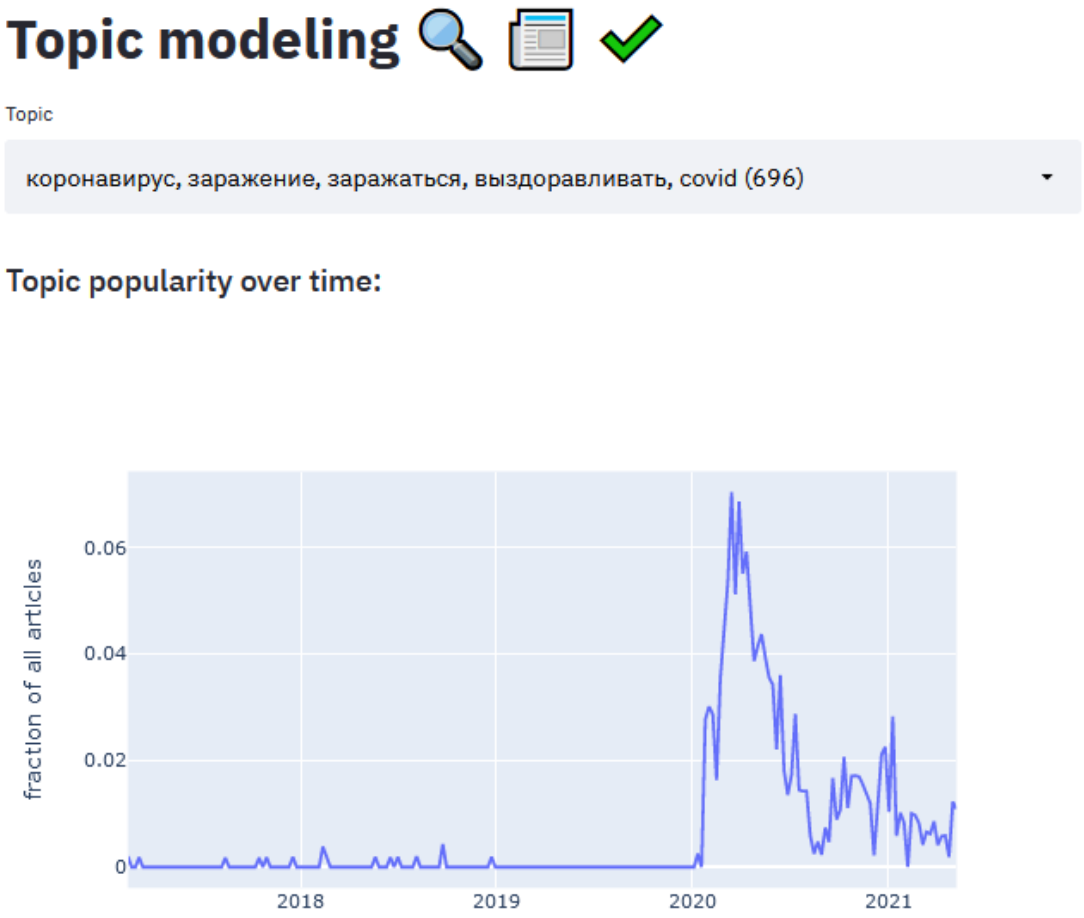



Для эбмеддингов BERT:



## Скриншоты сайта с визуализациями

https://share.streamlit.io/sinytim/topics



# Topic modeling 🔍 📰 ✅

Topic

коронавирус, заражение, заражаться, выздоравливать, covid (696)

Article

Евросоюз выходит из локдауна. В школу, в ресторан, в отпуск — уже можно?

### The most similar articles to the selected one:

*[2021-05-08 13:36:00]* **Евросоюз выходит из локдауна. В школу, в ресторан, в отпуск — уже можно?** [Евросоюз, Коронавирус, вакцинация, туризм] [коронавирус, заражение, заражаться, выздоравливать, covid] (distance: 0.000)

*[2020-06-06 09:33:00]* **Карантину конец? Изучили отчеты Google о передвижениях людей в Беларуси и других странах** [Коронавирус, Беларусь - Россия, Транспорт, эпидемии] [коронавирус, заражение, заражаться, выздоравливать, covid] (distance: 0.040)

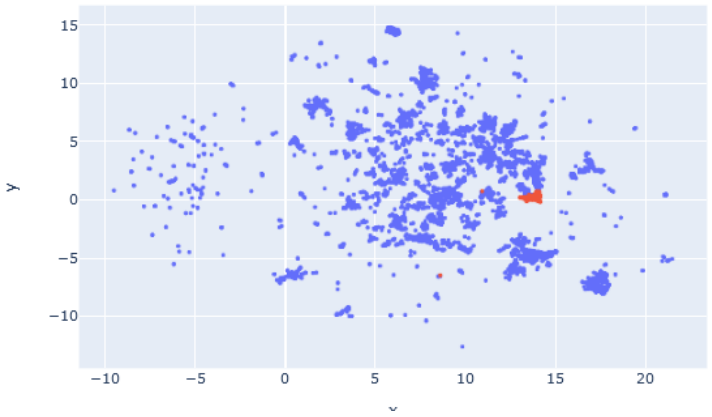
*[2020-04-24 06:20:00]* **Где в Европе уже начали смягчать карантин** [В мире, Коронавирус, дети, СМИ, торговля, эпидемии] [коронавирус, заражение, заражаться, выздоравливать, covid] (distance: 0.047)

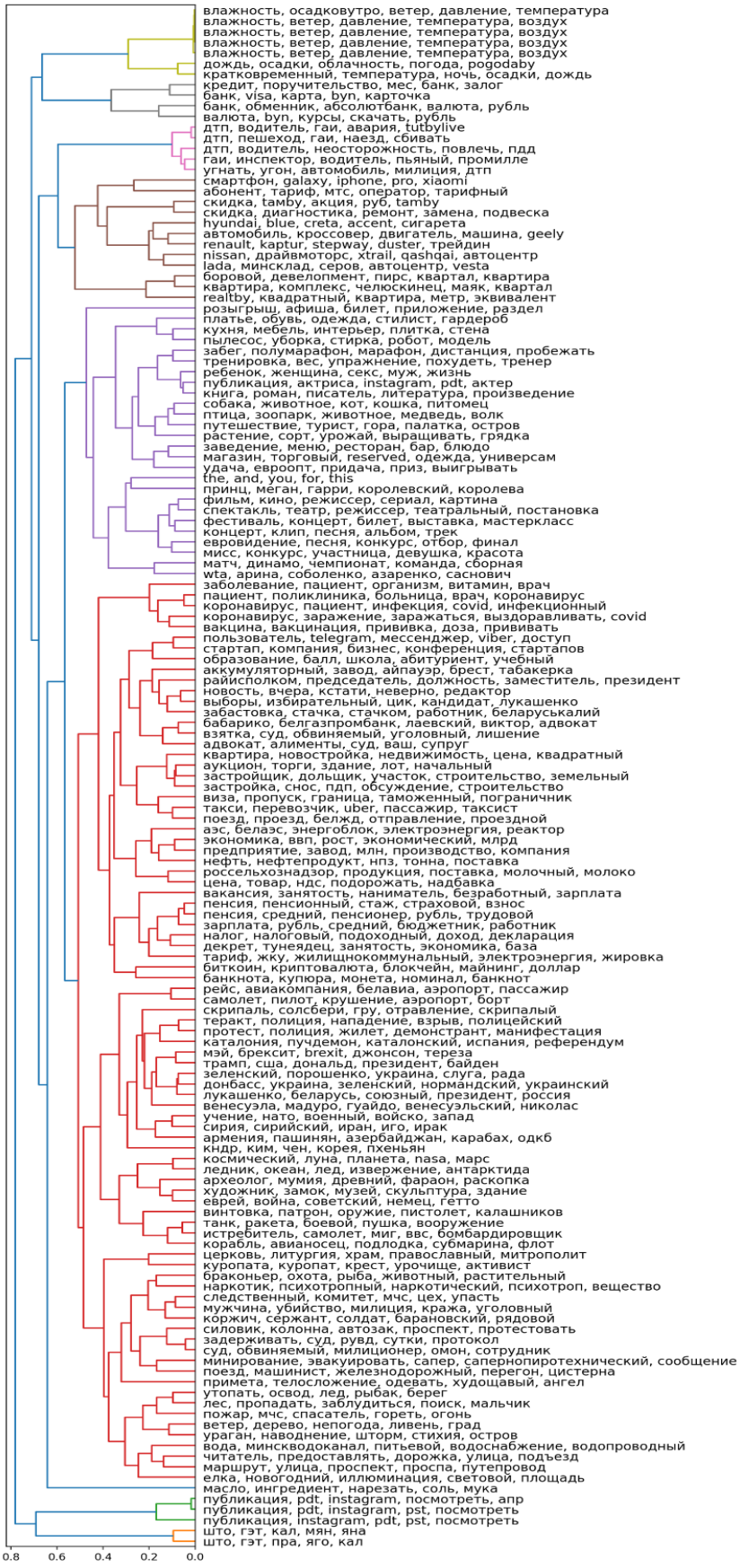
# Topic modeling 🔍 📰 ✅

Topic

коронавирус, заражение, заражаться, выздоравливать, covid (696)

### Each point is an article in a semantic space.




влажность, осадковутро, ветер, давление, температура
влажность, ветер, давление, температура, воздух
влажность, ветер, давление, температура, воздух
влажность, ветер, давление, температура, воздух
влажность, ветер, давление, температура, воздух
дождь, осадки, облачность, погода, pogodaby
кратковременный, температура, ночь, осадки, дождь
кредит, поручительство, мес, банк, залог
банк, visa, карта, byn, карточка
банк, обменник, абсолютбанк, валюта, рубль
валюта, byn, курсы, скачать, рубль
дтп, водитель, гаи, авария, tutbylive
дтп, пешеход, гаи, наезд, сбивать
дтп, водитель, неосторожность, повлечь, пдд
гаи, инспектор, водитель, пьяный, промилле
угнать, угон, автомобиль, милиция, дтп
смартфон, galaxy, iphone, pro, xiaomi
абонент, тариф, мтс, оператор, тарифный
скидка, tamby, акция, руб, tamby
скидка, диагностика, ремонт, замена, подвеска
hyundai, blue, creta, accent, сигарета
автомобиль, кроссовер, двигатель, машина, geely
renault, kaptur, stepway, duster, трейдин
nissan, драйвмоторс, xtrail, qashqai, автоцентр
lada, минсклад, серов, автоцентр, vesta
боровой, девелопмент, пирс, квартал, квартира
квартира, комплекс, челюскинец, маяк, квартал
realtby, квадратный, квартира, метр, эквивалент
розыгрыш, афиша, билет, приложение, раздел
платье, обувь, одежда, стилист, гардероб
кухня, мебель, интерьер, плитка, стена
пылесос, уборка, стирка, робот, модель
забег, полумарафон, марафон, дистанция, пробежать
тренировка, вес, упражнение, похудеть, тренер
ребенок, женщина, секс, муж, жизнь
публикация, актриса, instagram, pdt, актер
книга, роман, писатель, литература, произведение
собака, животное, кот, кошка, питомец
птица, зоопарк, животное, медведь, волк
путешествие, турист, гора, палатка, остров
растение, сорт, урожай, выращивать, грядка
заведение, меню, ресторан, бар, блюдо
магазин, торговый, reserved, одежда, универсам
удача, евроопт, придача, приз, выигрывать
the, and, you, for, this
принц, меган, гарри, королевский, королева
фильм, кино, режиссер, сериал, картина
спектакль, театр, режиссер, театральный, постановка
фестиваль, концерт, билет, выставка, мастеркласс
концерт, клип, песня, альбом, трек
евровидение, песня, конкурс, отбор, финал
мисс, конкурс, участница, девушка, красота
матч, динамо, чемпионат, команда, сборная
wta, арина, соболенко, азаренко, саснович
заболевание, пациент, организм, витамин, врач
пациент, поликлиника, больница, врач, коронавирус
коронавирус, пациент, инфекция, covid, инфекционный
коронавирус, заражение, заражаться, выздоравливать, covid
вакцина, вакцинация, прививка, доза, прививать
пользователь, telegram, мессенджер, viber, доступ
стартап, компания, бизнес, конференция, стартапов
образование, балл, школа, абитуриент, учебный
аккумуляторный, завод, айпауэр, брест, табакерка
райисполком, председатель, должность, заместитель, президент
новость, вчера, кстати, неверно, редактор
выборы, избирательный, цик, кандидат, лукашенко
забастовка, стачка, стачком, работник, беларуськалий
бабарико, белгазпромбанк, лаевский, виктор, адвокат
взятка, суд, обвиняемый, уголовный, лишение
адвокат, алименты, суд, ваш, супруг
квартира, новостройка, недвижимость, цена, квадратный
аукцион, торги, здание, лот, начальный
застройщик, дольщик, участок, строительство, земельный
застройка, снос, пдп, обсуждение, строительство
виза, пропуск, граница, таможенный, пограничник
такси, перевозчик, uber, пассажир, таксист
поезд, проезд, белжд, отправление, проездной
аэс, белаэс, энергоблок, электроэнергия, реактор
экономика, ввп, рост, экономический, млрд
предприятие, завод, млн, производство, компания
нефть, нефтепродукт, нпз, тонна, поставка
россельхознадзор, продукция, поставка, молочный, молоко
цена, товар, ндс, подорожать, надбавка
вакансия, занятость, наниматель, безработный, зарплата
пенсия, пенсионный, стаж, страховой, взнос
пенсия, средний, пенсионер, рубль, трудовой
зарплата, рубль, средний, бюджетник, работник
налог, налоговый, подоходный, доход, декларация
декрет, тунеядец, занятость, экономика, база
тариф, жку, жилищнокоммунальный, электроэнергия, жировка
биткоин, криптовалюта, блокчейн, майнинг, доллар
банкнота, купюра, монета, номинал, банкнот
рейс, авиакомпания, белавиа, аэропорт, пассажир
самолет, пилот, крушение, аэропорт, борт
скрипаль, солсбери, гру, отравление, скрипалый
теракт, полиция, нападение, взрыв, полицейский
протест, полиция, жилет, демонстрант, манифестация
каталония, пучдемон, каталонский, испания, референдум
мэй, брексит, brexit, джонсон, тереза
трамп, сша, дональд, президент, байден
зеленский, порошенко, украина, слуга, рада
донбасс, украина, зеленский, нормандский, украинский
лукашенко, беларусь, союзный, президент, россия
венесуэла, мадуро, гуайдо, венесуэльский, николас
учение, нато, военный, войско, запад
сирия, сирийский, иран, иго, ирак
армения, пашинян, азербайджан, карабах, одкб
кндр, ким, чен, корея, пхеньян
космический, луна, планета, nasa, марс
ледник, океан, лед, извержение, антарктида
археолог, мумия, древний, фараон, раскопка
художник, замок, музей, скульптура, здание
еврей, война, советский, немец, гетто
винтовка, патрон, оружие, пистолет, калашников
танк, ракета, боевой, пушка, вооружение
истребитель, самолет, миг, ввс, бомбардировщик
корабль, авианосец, подлодка, субмарина, флот
церковь, литургия, храм, православный, митрополит
куропата, куропат, крест, урочище, активист
браконьер, охота, рыба, животный, растительный
наркотик, психотропный, наркотический, психотроп, вещество
следственный, комитет, мчс, цех, упасть
мужчина, убийство, милиция, кража, уголовный
коржич, сержант, солдат, барановский, рядовой
силовик, колонна, автозак, проспект, протестовать
задерживать, суд, рувд, сутки, протокол
суд, обвиняемый, милиционер, омон, сотрудник
минирование, эвакуировать, сапер, сапернопиротехнический, сообщение
поезд, машинист, железнодорожный, перегон, цистерна
примета, телосложение, одевать, худощавый, ангел
утопать, освод, лед, рыбак, берег
лес, пропадать, заблудиться, поиск, мальчик
пожар, мчс, спасатель, гореть, огонь
ветер, дерево, непогода, ливень, град
ураган, наводнение, шторм, стихия, остров
вода, минскводоканал, питьевой, водоснабжение, водопроводный
читатель, предоставлять, дорожка, улица, подъезд
маршрут, улица, проспект, проспа, путепровод
елка, новогодний, иллюминация, световой, площадь
масло, ингредиент, нарезать, соль, мука
публикация, pdt, instagram, посмотреть, апр
публикация, pdt, instagram, pst, посмотреть
публикация, instagram, pdt, pst, посмотреть
што, гэт, кал, мян, яна
што, гэт, пра, яго, кал
0.8
0.6
0.4
0.2
0.0