**УДК 004.912**

# Улучшения алгоритма кластеризации новостного потока текстовых сообщений

# Improvements of the algorithm of clustering of the news stream of text messages


1. **Гапанюк Ю.Е. (Gapanyuk Yu.E.), доцент кафедры «Системы обработки информации и управления» МГТУ им. Н.Э. Баумана, gapyu@bmstu.ru**
2. **Чернобровкин С.В. (Chernobrovkin S.V.), аспирант кафедры «Системы обработки информации и управления» МГТУ им. Н.Э. Баумана, sergey.chernobrovkin@inbox.ru**
3. **Мялкин М.П. (Myalkin M.P.), аспирант кафедры «Системы обработки информации и управления» МГТУ им. Н.Э. Баумана, maxmyalkin@gmail.com**
4. **Опрышко А.В. (Opryshko A.V.), аспирант кафедры «Системы обработки информации и управления» МГТУ им. Н.Э. Баумана, alexopryshko@yandex.ru**
5. **Латкин И.И. (Latkin I.I.), аспирант кафедры «Системы обработки информации и управления» МГТУ им. Н.Э. Баумана, igor.latkin@outlook.com**
6. **Леонтьев А.В. (Leontiev A.V.), аспирант кафедры «Системы обработки информации и управления» МГТУ им. Н.Э. Баумана, alekseyl@list.ru**
7. **Ожегов Григорий Андреевич (Ozhegov G.A.), аспирант кафедры «Системы обработки информации и управления» МГТУ им. Н.Э. Баумана, grigory@ozhegov.name**


## 1. Введение. Описание проблем базового алгоритма

### 1.1. Проблема семантического анализа текстов

В рассмотренном в статье [1] алгоритме потоковой кластеризации новостей наилучшим образом себя показала метрика расстояния TF-IDF. Качество кластеризации с использованием этой метрики в некоторых случаях достигало 85%. Однако полученная модель является статистической, то есть основана только на частоте вхождения слов в тексты, поэтому, несмотря на то, что она показывает приемлемые для задачи результаты, она не может «понимать» семантику предложения, и следовательно не будет хорошо работать на семантически похожих текстах, в которых почти нет общих слов.

Примером таких текстов могут быть: «Глава МИД Британии планирует в ближайшее время приехать в Россию» и «Борис Джонсон собирается вскоре посетить Москву». Эти предложения говорят об одном и том же, однако, мера Жаккара для них будет равна нулю.

Такие сложные случаи текущая модель распознавать неспособна, следовательно, необходимо улучшить ее для понимания значения текстов.

### 1.2. Проблема стабильного времени обработки нового элемента потока

Кластеризация i-ой новости происходит с помощью выбора среди уже имеющихся кластеров ближайшего. Близость текста к кластеру оценивается путем построения векторов TF-IDF для рассматриваемого текста и каждого из текстов кластера. Выбирается минимальное расстояние среди текстов, оно и берется за дальность текста от кластера.

Очевидно, что в таком алгоритме время обработки i-ой новости будет пропорционально количеству уже рассмотренных новостей, так как каждый текст сравнивается попарно со всеми остальными. Это может быть сильно заметно при выборе косинусной меры векторов TF-IDF в качестве метрики расстояния. Время обработки новости в зависимости от количества рассмотренных новостей (итерации алгоритма) представлено на рис. 1.

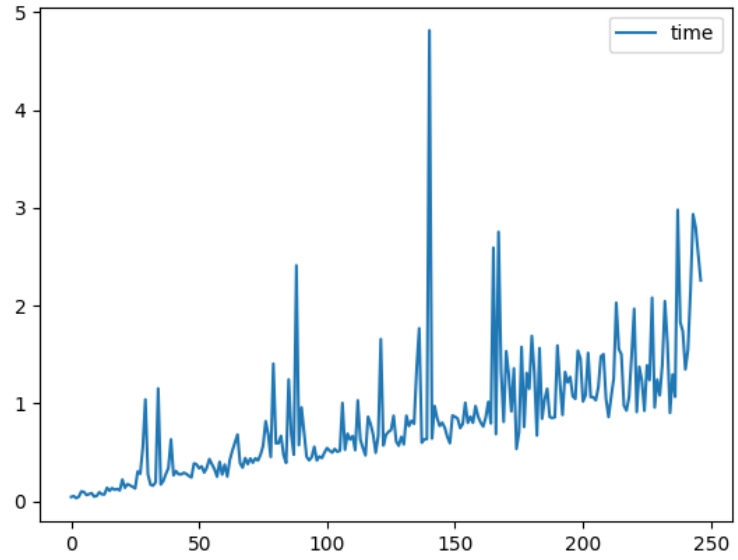


Рис. 1. Время обработки одной новости

На рис. 1 на оси X показан номер итерации, а на оси Y время обработки i-ой итерации. Видно, что время обработки i-ой новости растет с увеличением количества уже обработанных текстов. Общее время работы алгоритма для 247 новостей составляет 200 сек.

После 200-ой новости время каждой итерации становится более 1 секунды. Это является проблемой при постоянной длительной работе алгоритма.

## 2. Обзор способов использования семантики текста в задачах обработки текста

В работе [2] Томас Миколов и группа исследователей Google предложила модель векторного представления слов Word2Vec. Суть модели заключается в том, что каждому слову ставится в соответствие некоторый вектор таким образом, что вектора похожих по смыслу слов расположены в пространстве близко, а слова, несущие разный смысл - далеко. Таким образом, например, косинусная близость между словами “кофе” и “какао” может быть равна 0,7.

Попробуем использовать Word2Vec для улучшения качества кластеризации.

Word2Vec – это векторное представление слов в тексте, а не всего текста. Получается, что текст будет представлять собой матрицу:

$$Aij = \left(a_{ij}\right)^{mn},$$

где $m$ – количество слов в тексте; $n$ – размер вектора в word2vec.

Матрица $n$ x $m$ - слишком громоздкое представление текста, к тому же такие матрицы сложно сравнивать между собой. Для нашей задачи оптимальным решением было бы представить весь текст в виде одного вектора, чтобы использовать косинус между такими векторами в качестве метрики расстояния между текстами.

Чтобы перевести набор векторов слов в один вектор документа используем усреднение и взвешенное усреднение векторов слов.

### 2.1. Усреднение векторов слов

В качестве представления слова используем не вектор, а среднее арифметическое этого вектора. Тогда текст можно представить в виде набора средних арифметических значений векторов его слов.

$$V(d)=\{x_i\}, i=1..m,\ x_i=\sum_{j=1}^{n} v_j\ /n,$$

где $d$ – рассматриваемый документ; $w_j$ – значение в j-ой позиции вектора i-того слова.

На рис. 2 на оси X показан номер итерации, а на оси Y качество работы алгоритма с учетом данных, полученных до i-ой итерации.

На графике видно, что значение F-меры после стабилизации находится в окрестностях 0,73. Это значение хуже модели TF-IDF.

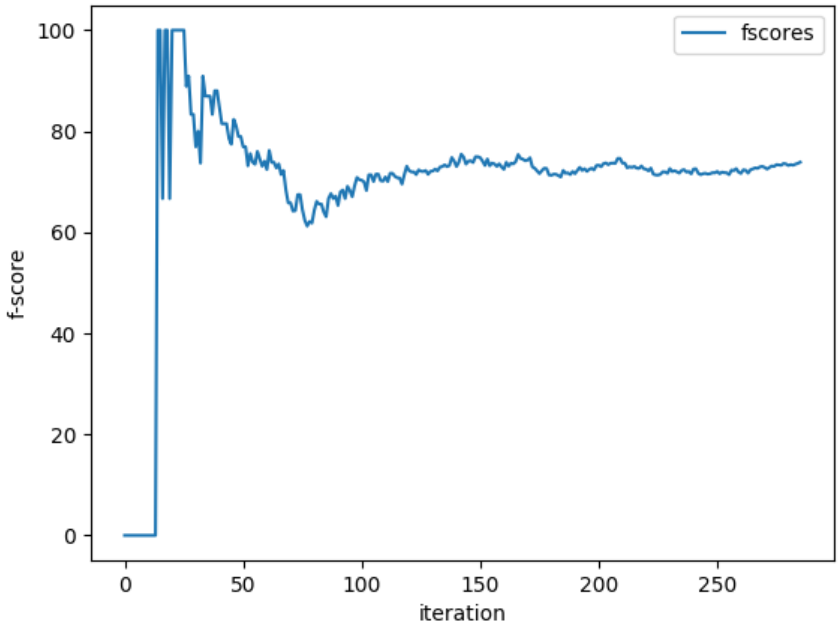


Рис. 2. F1-мера на каждой итерации для модели усредненного Word2Vec.

**2.2. Взвешенное усреднение векторов слов**

Описанная выше модель никак не учитывает частотность слов. Чтобы добавить ее в итоговый вектор документа, попробуем домножить каждое среднее значение слова на TF-IDF этого слова. Результаты экспериментов приведены на рис. 3.

$$x_i=\sum_{j=1}^{n} v_j\ /n * tfidf\left(w_i\right),\ \text{где}\ w_i\ \text{– i-ое слово.}$$

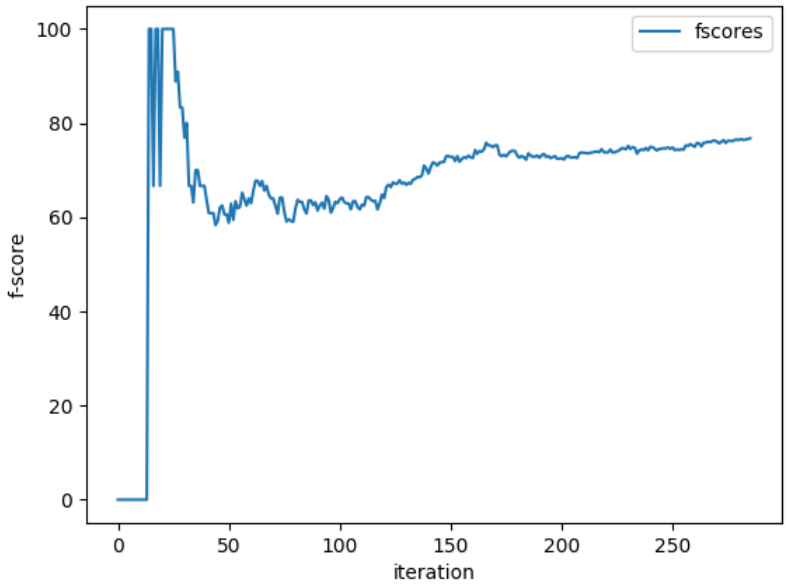


Рис. 3. F1-мера для модели взвешенных усредненных векторов Word2Vec.

Качество модели достигает 78%. Это на 5% лучше модели с усреднением векторов, однако по-прежнему хуже основной модели TF-IDF. Проблема заключается в усреднении вектора, ведь фактически после такой операции смысловая нагрузка слова размывается. Чтобы избавиться от этого недостатка рассмотрим еще одну метрику - Word Mover's Distance (WMD).

### 2.3. Использование WMD

В статье [3] М. Кушнир вводит понятие метрики расстояния между двумя текстами, основанное на Word2Vec. Суть WMD заключается в расширении задачи EMD (Earth Mover's Distance).

Формально задача определяется следующим образом. Распределение точек в пространстве можно представить в виде кластеров и их весов. Вес кластера – это отношение количества точек в кластере к общему числу точек. Такое представление распределение называется сигнатурой. Пусть задано 2 сигнатуры $P = \{(p1, w_{p1}), (p2, w_{p2}), ...\}$ и $Q = \{(q1, w_{q1}), (q2, w_{q2}), ...\}$, состоящие из кластеров $p_i$ и $q_i$ и их весов $w_{pi}$ и $w_{q1}$. Множества характеризуют набор товаров и их количество у «поставщика» и потребность в товарах и

необходимое количество у «потребителя». Задача состоит в минимизации целевой функции, то есть в нахождении оптимального потока $min\sum_{i=1}^{n}\sum_{j=1}^{m}f_{ij}d_{ij}$, где $f_{ij}$ – поток между $p_i$ и $q_i$; $d_{ij}$ – дистанция между $p_i$ и $q_i$. Тогда $\sum_{i=1}^{n}\sum_{j=1}^{m}f_{ij}=min\left(\sum_{i=1}^{m}w_{pi},\sum_{i=1}^{n}w_{qi}\right)$ – полный поток.

Зная оптимальный поток, метрику EMD можно посчитать следующим образом:

$$EMD(P,Q)=\frac{\sum_{i=1}^{n}\sum_{j=1}^{m}f_{ij}d_{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{m}f_{ij}}.$$

Для решения этой задачи было разработано множество решений. WMD берет свои корни в задаче нахождения EMD. В WMD дистанция между двумя словами i и j равна евклидовой норме разницы векторов этих слов в представлении word2vec. Веса $w_{pi}$ и $w_{qi}$ равны доле слов $p_i$ и $q_i$ в текстах P и Q соответственно.

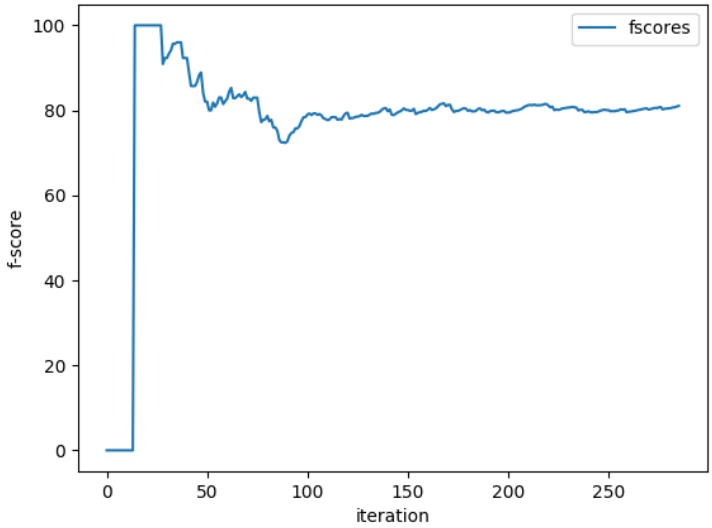


Рис. 4. Применение WMD для текстов.

Было поставлено несколько экспериментов с различными порогами. В лучшем случае качество модели 81% для случая на рис. 4. Это приблизительно на три процента меньше исходной модели с TF-IDF, однако, лучше взвешенного Word2Vec. Время на каждую итерацию в среднем равно 3.5сек, что в 4 раза больше исходного.

Качество полученной модели сильно зависит от результатов Word2Vec, количества слов в словаре.

**3. Предложенное решение по улучшению качества кластеризации**

Попробуем скомбинировать лучшие модели: в качестве метрики используем среднее арифметическое от WMD и расстояния по TF-IDF.

$$d(t_1, t_2) = \frac{EMD(t_1, t_2) + \cos(v(t_1), v(t_2))}{2}, \quad (1)$$

где $v(t_i)$ – вектор текста $t_i$, полученный по мере TF-IDF.

Результаты экспериментов представлены на рис. 5.

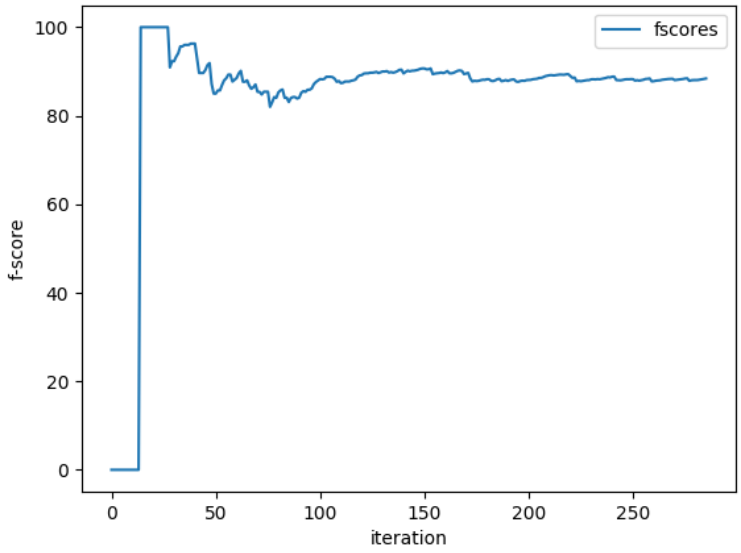


Рис. 5. Комбинация WMD + TF-IDF.

Возьмем медиану F-меры. Для WMD + TF-IDF она равна 88%, тогда как для TF-IDF на том же наборе данных составляет 86%. Таким образом, полученная модель улучшила исходный результат на 2%. Но время одной итерации составляет примерно ~6.5 сек.

Чтобы минимизировать время итерации, будем считать WMD не между текстами, а между заголовками новостей, результаты представлены на рис. 6.

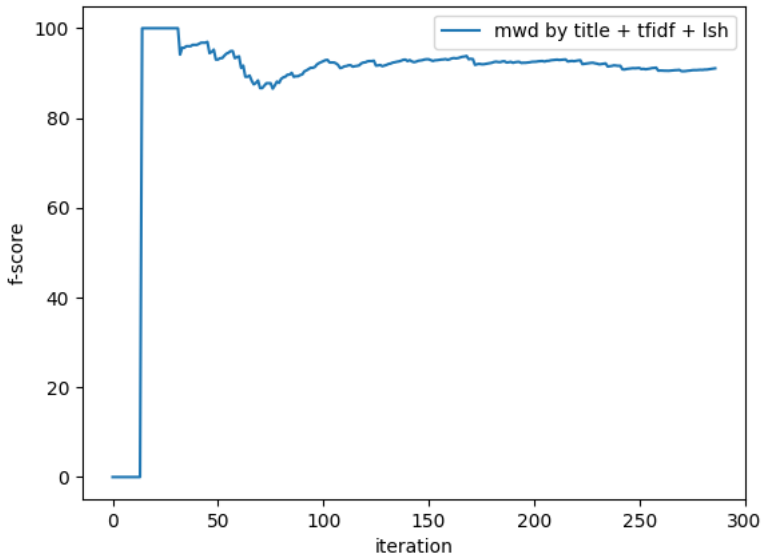


Рис. 6. Комбинация TF-IDF и WMD по заголовкам новостей.

Время одной итерации уменьшилось до 0,3 сек, а медиана F-меры достигла 92%. Это на 6% лучше исходной модели. Попробуем осуществить перемешивание данных и выполнить рассмотренные алгоритмы на основе Word2Vec несколько раз. Результаты экспериментов приведены в таблице 1 и на рис. 7.

Таблица 1. Результаты экспериментов для алгоритмов на основе Word2Vec.

| **№ эксперимента** | **TF-IDF** | **WMD + TF-IDF** | **Улучшение** |
|---|---|---|---|
| 1 | 81.0964647823 | 85.4580474299 | 4,361582648 |
| 2 | 76.0296144579 | 76.2838647806 | 0,254250323 |
| 3 | 82.5448154158 | 83.0170664757 | 0,47225106 |
| 4 | 86.2595419847 | 83.8926114536 | -2,366930531 |
| 5 | 84.2005012531 | 93.3034373686 | 9,102936116 |
| 6 | 76.6353441284 | 83.9413919414 | 7,306047813 |
| 7 | 80.0769271439 | 73.5700543951 | -6,506872749 |
| 8 | 82.9633408387 | 89.5718864469 | 6,608545608 |
| 9 | 79.8055479598 | 83.6250434387 | 3,819495479 |
| 10 | 77.3569272806 | 81.6214369492 | 4,264509669 |

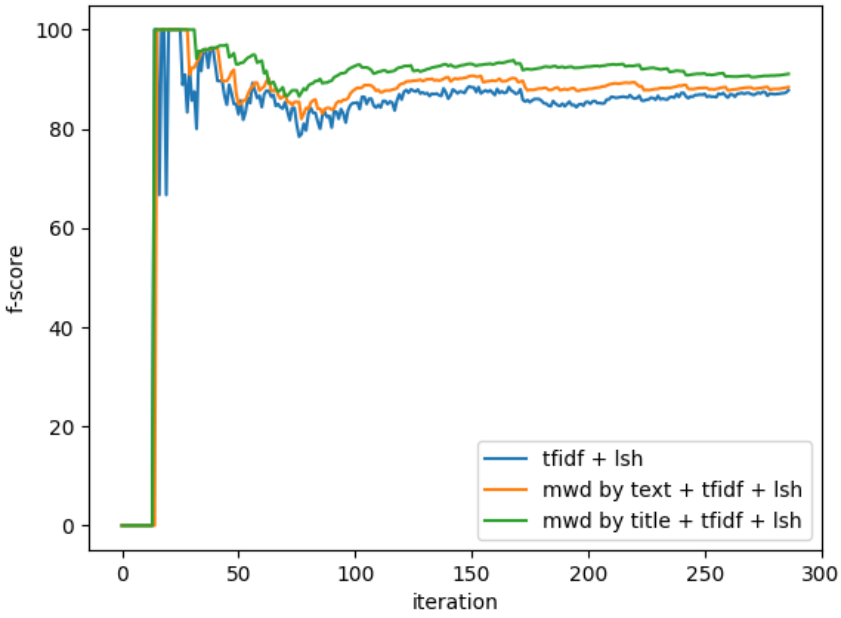


Рис. 7. Результаты экспериментов для алгоритмов на основе Word2Vec.

В 8 из 10 экспериментов результаты полученной модели лучше результатов исходной. В среднем результаты модели WMD + TF-IDF лучше на 2,7%.

**4. Обзор способов аппроксимации меры Жаккара между текстами**

В этом разделе рассмотрим основные способы стабилизации работы алгоритма кластеризации. На каждой итерации алгоритма основное время занимает обход всех существующих кластеров и их элементов, поэтому с увеличением числа кластеров и объемов данных растет и время обработки новых текстов. Это недопустимо, так как система должна обрабатывать новости онлайн и выполнять кластеризацию как можно быстрее.

Чтобы не проходить все кластеры заново при поступлении нового элемента применяется метод LSH.

Основная идея состоит в том, чтобы построить такие сигнатуры данных, которые были бы похожи друг на друга, если исходные данные также похожи между собой, и наоборот, сильно различались, если исходные данные различны. В качестве hash-функций будут использоваться и сравниваться simhash и minhash.

**4.1. Использование SimHash**

Алгоритм построения simhash рассмотрен в [4]. Краткое описание работы алгоритма:

1. Выбирается размер hash-значения, например 64 бита.
2. Формируется вектор из 64 ячеек, заполненный нулями.
3. Исходный текст разделяется на признаки, например нормализованные слова или н-граммы.
4. Каждый признак хешируется стандартным алгоритмом, например md5.
5. Для каждого хеша: Если i-ый бит = 1, то в выходном векторе в i-ую ячейку добавляется 1, в противном случае, вычитается.
6. Итоговый хеш для текста формируется следующим образом: если значение i-ой ячейки больше нуля, то i-ый бит хеша равен 1, в противном случае равен 0.

Итоговые хеши для похожих текстов будут похожи, то есть расстояние Хемминга между ними будет небольшим.

Таким образом, сформировав simhash для i-го текста необходимо найти из всего множества уже сформированных хешей те, расстояние Хемминга до которых будет меньше предопределенного порога. Дальше такие тексты необходимо сравнить с текущим с помощью метрики расстояния, как это делается в исходном предложенном алгоритме.

Такой алгоритм аппроксимирует дальность текста от кластера, однако, очевидно, что качество итоговой кластеризации может быть меньше, так как некоторые «правильные» тексты могут быть отсечены от сравнения еще на этапе аппроксимации.

Просто построить сигнатуры недостаточно, необходимо еще эффективно находить наиболее похожие на запрошенную сигнатуры среди всех остальных, то есть построить некоторый индекс по сигнатурам.

Таким образом, нужно решить следующую задачу: имея сигнатуру некоторого текста (запрос) необходимо найти все сигнатуры, отличающиеся побитово от запроса не более, чем на k бит.

Решать задачу полным перебором неэффективно. Даже просто отсортировав все сигнатуры, мы получим наборы подряд идущих сигнатур, значения которых будут различаться только в младших битах. Необходимо будет проверить их напрямую. Однако что делать со старшими битами? Если сигнатуры различаются только в старших битах, то в отсортированном списке такие сигнатуры будут располагаться далеко друг от друга. Чтобы избавиться от этой проблемы можно «повернуть» все сигнатуры, то есть сместить все биты на 1 влево. Тогда старший бит в исходной сигнатуре станет младшим в «смещенной». Далее необходимо снова отсортировать сигнатуры и выявить новые похожие. И так далее.

В статье [5] предлагается быстрый и эффективный алгоритм реализации подобной идеи. Результаты кластеризации с применением SimHash показаны на рис. 7.

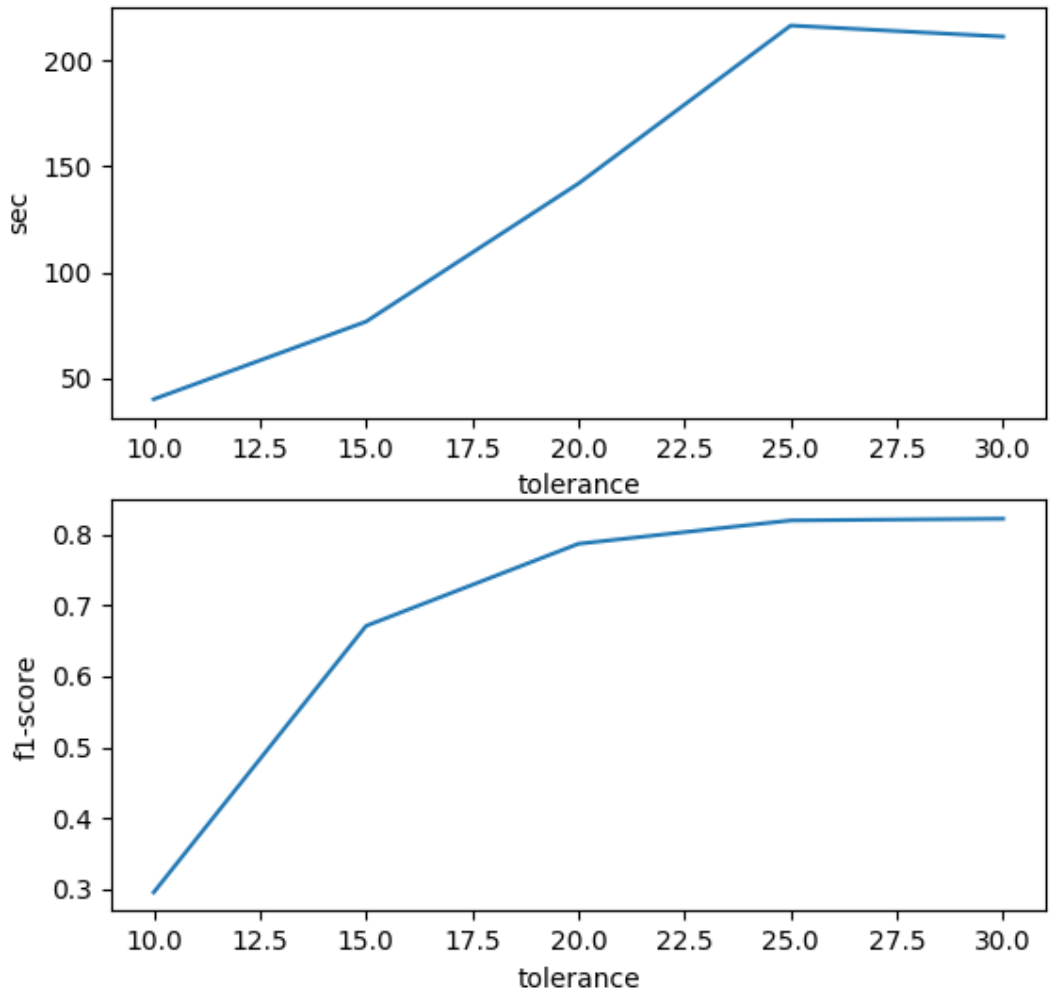


Рис. 7. Результаты кластеризации с применением SimHash.

На рис. 7 на оси X показан порог схожести сигнатур, а на оси Y качество алгоритма и время его работы при заданном пороге.

На графике видно, что, варьируя параметр порога схожести сигнатур, мы можем получать различные показатели эффективности алгоритма. Так, если порог слишком мал, то качество кластеризации, как и время работы, также слишком мало. Если же поднимать порог, то качество кластеризации и время работы будет расти.

Полученный результат может быть признан удовлетворительным. При значении порога 20 достигается приемлемое качество кластеризации (78% против исходных 82%), а время работы существенно снижается (с 200 сек до 140 сек).

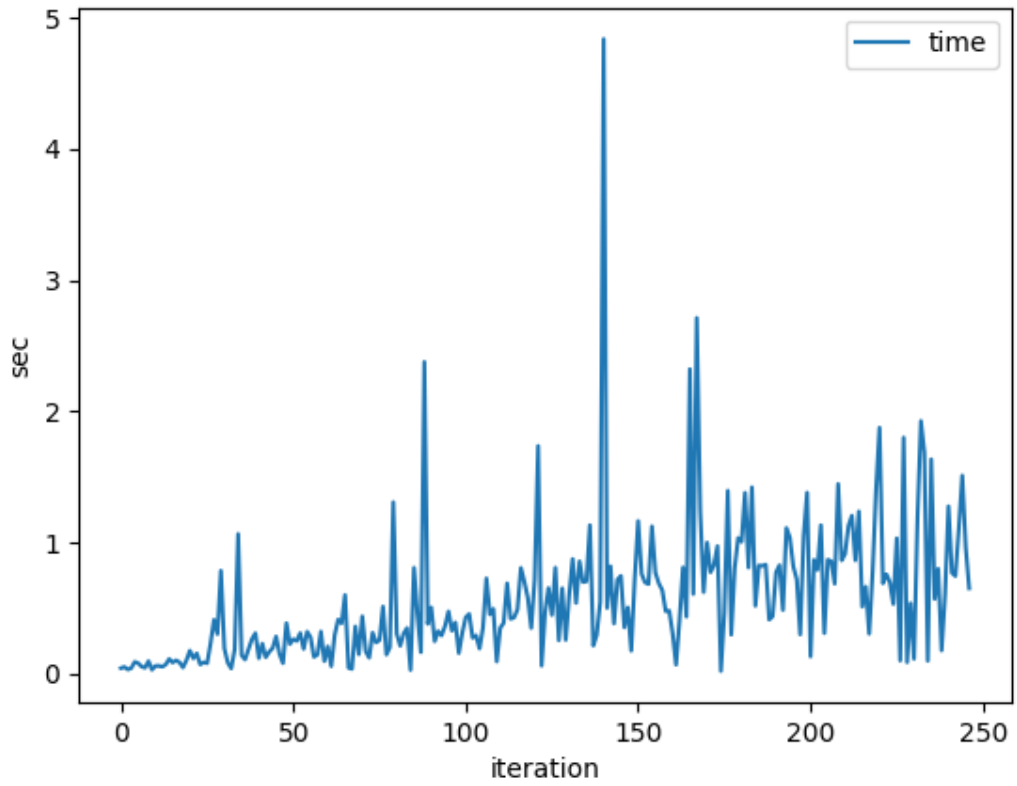


Рис. 8. Время на одну итерацию с использованием simhash.

При этом на рис. 8 показано, что время обработки одной новости не увеличивается, а находится в окрестностях 1 секунды.

### 4.2. Использование MinHash

Еще один алгоритм получения сигнатур minhash рассмотрен в [6]. Он используется для аппроксимации значения расстояния Жаккара между текстами.

Проблема сравнения текстов состоит в том, что пространство признаков текстов очень разреженно. Представим тексты в виде следующей характеристической матрицы (таблица 2).

Таблица 2. Характеристическая матрица представления текстов.

| | A B C | A C D | B D E |
|---|---|---|---|
| A | 1 | 1 | 0 |
| B | 1 | 0 | 1 |
| C | 1 | 1 | 0 |
| D | 0 | 1 | 1 |
| E | 0 | 0 | 1 |

Каждая строка матрицы соответствует элементу алфавита множества признаков текстов (например алфавитному слову). Каждый текст представляется вектором из 0 и 1. Единица ставится в i-ую позицию вектора, если в тексте присутствует i-ый элемент алфавита. Очевидно, что если текстов много и они достаточно разнородные, то все векторы будут состоять преимущественно из единиц. MinHash – техника составления сигнатур фиксированного размера для векторов текстов, причем таких, что, если исходные векторы похожи, то и их хеши должны быть похожи.

Для подсчета minhash используют следующую процедуру: необходимо перемешать строки характеристической матрицы и для каждого столбца найти первую позицию, где встретится единица. Повторяя такую процедуру N раз, можно составить вектор текста размером N.

Доказано, что вероятность того, что minhash для случайного «перемешивания» строк сгенерирует одинаковые значения для 2-х множеств равна мере Жаккара между этими

множествами. Соответственно, minhash можно использовать для аппроксимации расстояния между двумя множествами.

Существует алгоритм эффективного нахождения сигнатур без явного перемешивания матрицы.

Однако в нашей задаче необходимо находить хеши, не имея других данных, то есть не зная алфавита. Для этого используем классическую процедуру формирования minhash:

1. Разобьем текст на слова.
2. Хешируем каждое слово с помощью функции хеширования.
3. Находим минимальное значение хеша, полученное на втором шаге.
4. Повторяя процедуру N раз с разными хеш-функциями получаем вектор размером N.

После составления сигнатуры для i-того текста необходимо подобрать наиболее похожие сигнатуры, чтобы уже напрямую сравнить тексты с помощью любой из метрик.

Такая техника называется LSH - Locality-sensitive hashing [7]. Рассмотрим, как это можно применить для задачи кластеризации документов.

### 4.3. Использование LSH

Основной подход LSH – хешировать значения несколько раз, чтобы похожие значения попадали в одинаковые ячейки. Значения в каждой ячейке являются «кандидатами» на более точное сравнение. Считается, что доля false-positives (непохожих значений, попавших в одну ячейку) и false-negatives (похожих значений, попавших в разные ячейки) будет небольшой.

Применить LSH в задаче можно следующим образом:

1. Составим матрицу из minhash-сигнатур для каждого текста. Каждый столбец матрицы - вектор minhash-ей для текста.
2. Разобьем матрицу на b диапазонов по r строк.

3. Для каждого столбца из r элементов в каждом из диапазонов применим hash-функцию. Причем для каждого диапазона набор ячеек должен быть разным.

Таким образом, получим множество ячеек хеш-таблицы. Если сигнатуры двух текстов одинаковы хоть в каком-нибудь из диапазонов, то, по определению хеш-функции, они попадут в одинаковую ячейку, а значит, окажутся кандидатами на сравнение.

Рассмотрим более детально метод разбиения на диапазоны. В приведенных далее формулах $b$ – количество диапазонов; $r$ – количество строк в диапазоне; $s$ – мера Жаккара между двумя текстами.

Вероятность того, что две сигнатуры совпадают во всех строках в одном диапазоне равна $s^r$. Вероятность того, что две сигнатуры не совпадают хотя бы в одной строке в одном диапазоне равна $1-s^r$. Вероятность того, что две сигнатуры не совпадают во всех строках во всех диапазонах равна $(1-s^r)^b$. Вероятность того, что две сигнатуры совпадают во всех строках хотя бы в одном из диапазонов равна $1-(1-s^r)^b$. Вероятность того, что два текста попадут в «кандидаты на сравнение» равна $1-(1-s^r)^b$. На рис. 8 видно, что вне зависимости от выбранных r и b график принимает вид S-кривой, где порог роста можно определить по формуле $t=(1/b)^{1/r}$.

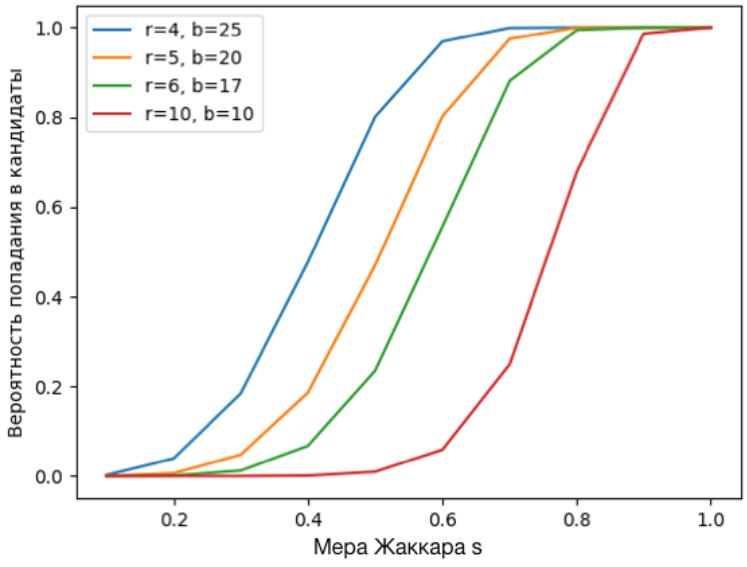

Рис. 8. Зависимость вероятности попадания в «кандидаты на сравнение» от r и b.

На графиках, приведенных на рис. 9, показано, что при увеличении порога необходимой «схожести» по Жаккару, время работы всего алгоритма, как и его точность, уменьшаются. Это можно объяснить тем, что чем больше порог, тем меньше текстов попадает в одну хеш-ячейку и тем меньше нужно времени на их проверку, но при этом растет количество false-negative примеров, что ведет к ухудшению качества.

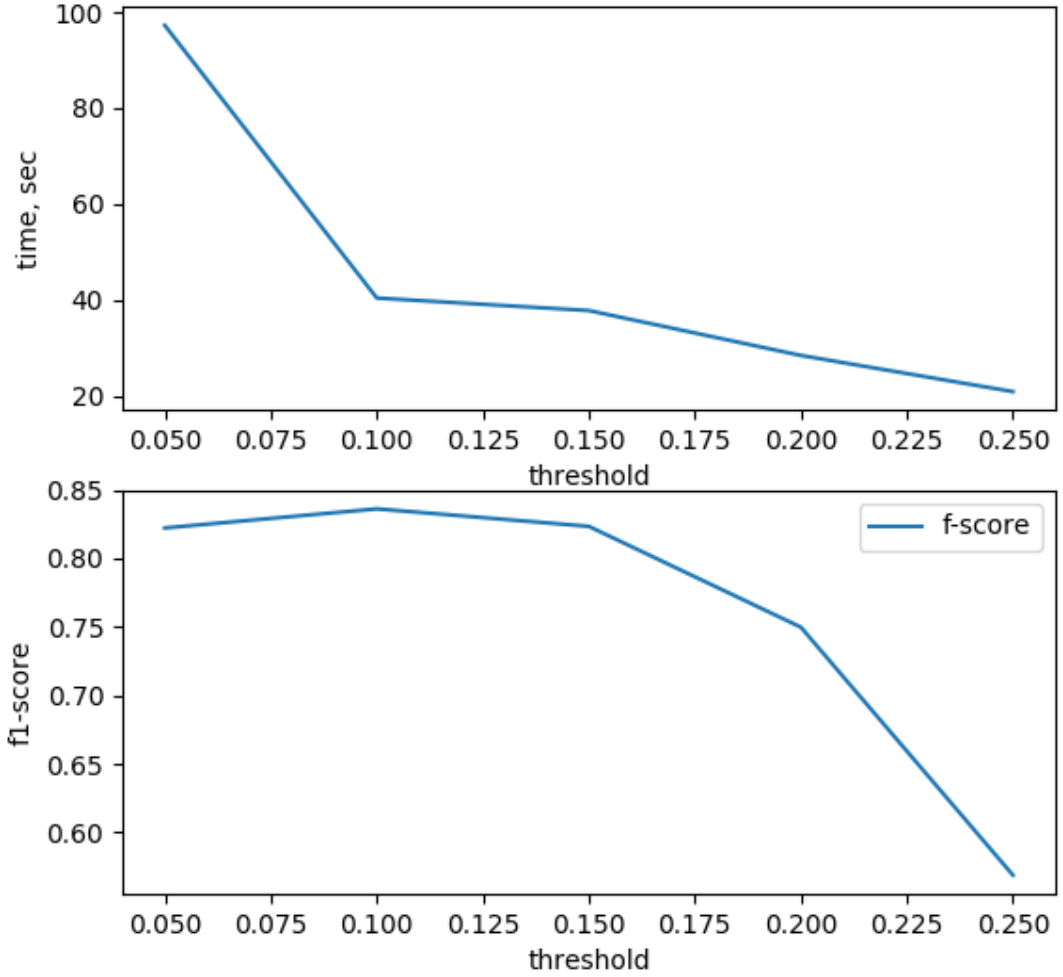


Рис. 9. Применение MinHash и LSH. Общее время работы.

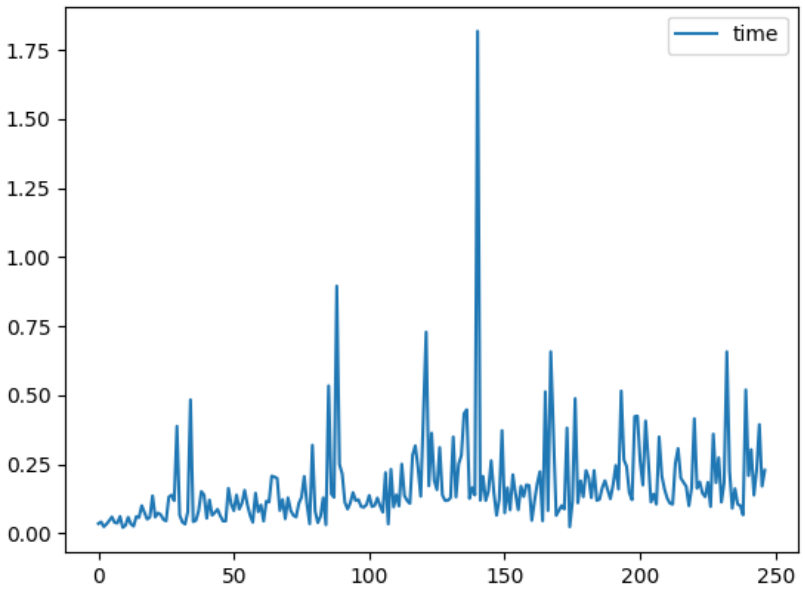


Рис. 10. Время итерации при использовании MinHash и LSH.

На рис. 10 на оси X отмечен номер итерации, а на оси Y – время обработки алгоритмом данных, поступивших на i-ой итерации. Как видно из рис. 10, время работы алгоритма не зависит от количества предварительно кластеризованных новостей. Время является константой и составляет всего около 0.25 секунд, что значительно меньше времени работы алгоритма без применения рассмотренных методов.

**5. Обобщенный улучшенный алгоритм кластеризации новостного потока**

Обобщим результаты проведенных экспериментов и сформулируем улучшенный алгоритм кластеризации новостного потока.

В статье [1] была предложена структура системы кластеризации новостного потока текстовых сообщений на основе метаграфового подхода [8]. Основным элементом системы является модуль кластеризации, который может быть представлен в виде метаграфового агента.

Метаграфовый агент работает на основе следующей системы правил. Вначале с использованием алгоритма кластеризации *Alg* для нового сообщения осуществляется поиск наиболее подходящего кластера. Далее выполняется одно из двух следующих правил:

1. если наиболее подходящий кластер найден, то новая вершина добавляется в данный кластер;
2. если наиболее подходящий кластер не найден, то создается новый кластер и новая вершина добавляется в созданный кластер.

При этом кластеры и сообщения представляются в виде элементов метаграфовой модели, также при необходимости могут формироваться вспомогательные внутрикластерные и межкластерные ребра, используемые для работы алгоритма.

Подход, предложенный в данной статье, не меняет ни структуры системы кластеризации новостного потока текстовых сообщений, ни системы правил метаграфового агента модуля кластеризации. Единственным модифицированным элементом является алгоритм кластеризации *Alg*.

Напомним, что в соответствии с [1] алгоритм кластеризации *Alg* состоит из двух шагов.

Шаг 1. $Dist_{\min} = \min\left(Dist\left(MVC_k, v_i\right)\right), \forall MVC_k \in MVC$

Для каждого кластера $MVC_k$, входящего во множество кластеров *MVC* на основе метрики *Dist* вычисляется расстояние между кластером и новой вершиной-сообщением $v_i$. Из вычисленных расстояний выбирается минимальное расстояние $Dist_{\min}$.

Шаг 2. $MVC_{RES} = \begin{cases} Dist_{\min} < Dist_{\text{threshold}} \rightarrow MVC_k \\ Dist_{\min} \geq Dist_{\text{threshold}} \rightarrow \varnothing \end{cases}$

Выходной кластер $MVC_{RES}$ определяется на основе порогового расстояния $Dist_{\text{threshold}}$. Если найденное минимальное расстояние меньше порогового, то в качестве найденного кластера выбирается кластер $MVC_k$, соответствующий найденному минимальному расстоянию. Если найденное минимальное расстояние больше порогового, то возвращается признак того что кластер не найден, что приводит к созданию нового кластера.

Пороговое расстояние $Dist_{\text{threshold}}$ и метрика *Dist* являются гиперпараметрами алгоритма *Alg*.

В качестве метрики *Dist* статье [1] было предложено использовать коэффициент Жаккара и меру TF-IDF. Приведенные в [1] результаты экспериментов показали, что мера TF-IDF обеспечивает лучшее качество работы алгоритма.

Но, как показали результаты экспериментов, приведенные в данной статье, подход [1] обладает следующими недостатками:

- Недостаток 1. Подход [1] не решает задачи сходства семантики текстов, из-за этого семантически близкие новости, выраженные различными словами, могут попасть с разные кластеры.
- Недостаток 2. Поскольку значение $Dist\left(MVC_k, v_i\right)$ вычисляется методом перебора для всех кластеров, то время работы алгоритма кластеризации растет пропорционально количеству накопленных новостей.

Для исправления отмеченных недостатков предлагается алгоритм *Alg'*. Шаг 2 в алгоритмах *Alg* и *Alg'* совпадает, а шаг 1 алгоритма *Alg* разбивается на несколько шагов в алгоритме *Alg'*. Рассмотрим более детально шаг 1 алгоритма *Alg'*.

Шаг 1 состоит из следующих шагов:

Шаг 1.1. $v_{Minhash} = Minhash\left(v_i\right)$.

Для кластеризуемой новости $v_i$ вычисляется значение minhash. Отметим, что рассчитанное значение впоследствии будет сохранено вместе с новостью в соответствующий кластер, поэтому для всех кластеризованных новостей значение $v_{Minhash}$ уже известно.

Шаг 1.2. $MVC_k^{SEL}\left(vc_j^{SEL}\right) = LSH\left(MVC_k\left(vc_j\right), v_i\right), \forall vc_j \in MVC_k, \forall MVC_k \in MVC$.

Для каждой ранее кластеризованной новости $vc_j$, принадлежащей кластеру $MVC_k$, входящего во множество кластеров *MVC,* и кластеризуемой новости $v_i$, применяется процедура LSH на основе ранее рассчитанных значений minhash. В результате применения процедуры LSH формируется множество выбранных элементов $vc_j^{SEL}$, входящих во множество выбранных кластеров $MVC_k^{SEL}$. Выбранные элементы $vc_j^{SEL}$ являются элементами, наиболее подходящими для сравнения с кластеризуемой новостью $v_i$. Количество выбранных элементов может быть существенно меньше, чем общее количество элементов. В множестве выбранных кластеров $MVC_k^{SEL}$ остаются только те кластеры, которые содержат выбранные элементы.

Шаги 1.1 и 1.2 позволяют преодолеть недостаток 2 и увеличивают скорость работы алгоритма кластеризации, так как в кластеризации используется «прореженное» множество выбранных элементов.

Шаг 1.3. $Dist_{\min} = \min\left(Dist\left(MVC_k^{SEL}\left(vc_j^{SEL}\right), v_i\right)\right)$.

Вычисляется расстояние для всех выбранных элементов $vc_j^{SEL}$ и кластеризуемой новости $v_i$. В качестве метрики *Dist* используется среднее арифметическое между WMD от заголовков статей и косинусного расстояния TF-IDF векторов содержимого статьей в соответствии с формулой (1). Из всех выбранных расстояний находится минимальное расстояние $Dist_{\min}$, которое далее используется на втором шаге алгоритма.

Шаг 1.3 позволяет преодолеть недостаток 1 и учесть при кластеризации сходство семантики текстов новостей за счет использования метрики WMD.

Таким образом, шаги 1.1, 1.2 и 1.3 алгоритма *Alg'* как и шаг 1 алгоритма *Alg* позволяют найти минимальное расстояние $Dist_{\min}$. Но при этом предложенный подход позволяет устранить выявленные недостатки 1 и 2.

**Выводы**

В статье рассмотрены способы улучшения семантической составляющей алгоритма кластеризации потока новостей и стабилизации времени обработки поступающих данных.

Была предложена метрика расстояния, комбинирующая лучшие свойства рассмотренных метрик, а также обновлен алгоритм с учетом предложенной метрики.

В итоге получен новый, улучшенный алгоритм кластеризации новостного потока, поставлены эксперименты по его сравнению с исходной версией. Улучшение по качеству кластеризации составляет около 2.7%, время обработки нового элемента сокращается примерно в четыре раза.

**Литература**

1. Гапанюк Ю.Е., Чернобровкин С.В., Латкин И.И., Леонтьев А.В., Ожегов Г.А., Опрышко А.В., Мялкин М.П. Алгоритм кластеризации новостного потока текстовых сообщений. Информационно-измерительные и управляющие системы. 2017. Выпуск № 10. С. __-__.

2. T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean. Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 2013, pp. 3111-3119.

3. Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, Kilian Q. Weinberger. From Word Embeddings To Document Distances. ICML'15 Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, 2015, pp. 957-966.

4. M. Charikar. Similarity estimation techniques from rounding algorithm. ACM Symposium on Theory of Computing, 2002, pp. 380-388.

5. Gurmeet Singh Manku, Arvind Jain, Anish Das Sarma. Detecting Near-Duplicates for Web Crawling. Proceedings of the 16th international conference on World Wide Web, WWW '07, 2007, pp. 141-150.

6. A. Shrivastava, P. Li. In Defense of MinHash Over SimHash. Proceedings of the 17-th International Conference on Artificial Intelligence and Statistics (AISTATS), 2014, pp. 886-894.

7. Hongya Wang, Jiao Cao, LihChyun Shu, Davood Rafiei. Locality sensitive hashing revisited: filling the gap between theory and algorithm analysis. Proceedings of the 22nd ACM international conference on Information & Knowledge Management CIKM '13, 2013, pp. 1969-1978.

8. Черненький В.М., Терехов В.И., Гапанюк Ю.Е. Структура гибридной интеллектуальной информационной системы на основе метаграфов. Нейрокомпьютеры: разработка, применение. 2016. Выпуск №9. С. 3-14.