Титульный лист

Содержание

1 Введение

В современном мире информация быстро устаревает, поэтому способы вовремя находить нужные данные — постоянный объект для исследований. Одним из направлений в этой области является извлечение данных из микроблоггинговых платформ.

Платформы микроблогов стали очень популярным способом размещения данных в Сети. В них можно найти сообщения пользователей практически на любую тему, начиная стихийными бедствиями и заканчивая рейтингами музыкальных исполнителей. Правильная обработка доступной информации — нетривиальная задача, которая имеет множество областей применения. Последние несколько лет эта тема активно исследуется во многих университетах мира.

Отслеживание сообщений о стихийных бедствиях в реальном времени поможет вовремя организовать спасательные операции и сохранить жизни людей[1]. Руководствуясь сообщениями пользователей микроблогов можно судить о популярности товаров и вовремя принимать экономически целесообразные решения. Можно делать предположения о рейтингах политических деятелей и эффективности рекламы на основании информации в микроблогах. Помимо перечисленных способов применения доступной информации в микроблоггинговых платформах можно привести множество других.

В данной работе в дальнейшем будет рассматриваться сервис микроблогов Твиттер[1]. В нем помимо текстовой информации можно публиковать фото, видео и геотеги, что так же может быть использовано при анализе, но в этой работе не рассматривается. В доступном наборе данных можно проводить анализ разных сущностей, эта работа посвящена выявлению событий среди потоков информации. Поскольку трактовка событий в сообщениях микроблогов может быть субъективной, выделим несколько свойств, которыми характеризуется событие.

Событие в первую очередь должно быть чем-то аномальным на фоне остальных данных. Оно определяется резким изменением частотных характеристик некоторых слов в сообщениях. События в микроблогах носят взрывной характер, в течении нескольких часов частоты релевантных слов возрастают в десятки раз и так же быстро опускаются до нормального уровня. Примером события могут быть: стихийное бедствие, выход спорного законопроекта на резонансную тему, получение фильмом награды на кинофестивале.

Исторически подходы к описанной задаче менялись, в следующем разделе будет рассмотрена формальная постановка задачи и эволюция способов ее решения.

2 Описание задачи

Цель данной работы состоит из нескольких частей:

- исследовать существующие подходы по извлечению событий из сети Твиттер,
- исследовать возможность применения тематических моделей для решения описанной задачи,
- разработать метод для извлечения событий на основе иерархического процесса Дирихле,

[1] http://www.twitter.com

- продемонстрировать работу алгоритма на реальных данных.

Объект изучения этой работы — алгоритм, который по входным данным строит множество событий. В качестве данных для задач подобного рода служит мультимножество документов (сообщений):

$$\Omega = \left\{ D_i \mid i \in \overline{1,n} \right\}. \tag{1}$$

Будем считать, что каждый документ имеет временную метку t. Документ D_i определяется как упорядоченный набор слов:

$$D_i = \left\{ w_j \mid j \in \overline{1,l_i} \right\}, \tag{2}$$

при этом слова в документах принадлежат некоторому словарю V.

Событие — некоторая сущность, которая характеризуется временем возникновения и ключевыми словами. Оно вызывает резкий подъем частотных характеристик некоторых слов. Событием может быть футбольный матч и музыкальный концерт. В социальной сети Твиттер есть популярные темы, которые всегда содержат много сообщений. Например это сообщения с ключевыми словами iphone и ipad. Но такие сообщения нельзя считать событиями. Также событиями нельзя считать еженедельные пятничные сообщения о конце рабочей недели[2].

Особенности социальной сети Твиттер состоят в следующем:

- короткие сообщения (до 140 символов),
- наличие шума и ошибок,
- большая плотность сообщений.
- взрывной характер событий,

3 Анализ существующих подходов

Прежде чем составлять решение задачи были изучены уже существующие подходы. Все они комбинируют техники машинного обучения, обработки текстов, вероятностных графических моделей и других разделов науки. Разные методы хорошо работают для одних данных и плохо для других, на их качество также сильно влияет природа данных. Выделим несколько подходов, для того чтобы показать общую схему подобных алгоритмов и чтобы выделить идеи, повлиявшие на алгоритм, предложенный автором этой работы.

сделать введение и заключение к каждому методу

3.1 NED

Исторически первым подходом к извлечению событий принято считать NED (New Event Detection)[3]. NED предназначен для того, чтобы находить первый документ на тему, которая не встречалась раньше. Следующие документы на эту тему уже не будут новыми и не будут помечены алгоритмом. Для того, чтобы отвечать на вопрос, является ли документ новым, необходимо указать способ как определять степень сходства двух документов.

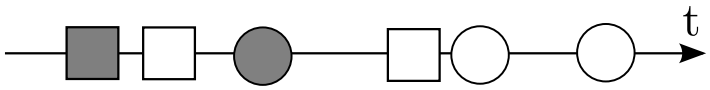

Рис. 1: Документы, соответствующие двум разным темам. Новые документы помечены серым цветом.

Для этого в алгоритме NED используется техника Incremental TF–IDF (Term Frequency – Inverse Document Frequency). TF–IDF — базовый метод выяснять насколько отдельно взятые слова характеризуют весь документ, другими словами, насколько большой вес имеет слово w в документе d. Пусть $f(d, w)$ — количество слов w в документе d. Определим значение $df_t(w)$ как количество документов, поступивших не позднее времени t, в которых встречается слово w. Используя введенные величины, можно записать значение веса определенного слова w в документе d. В момент времени t имеем:

$$\text{weight}_t(d, w) = \frac{1}{Z_t(d)} f(d, w) \cdot \log \frac{N_t}{df_t(w)}, \tag{3}$$

где N_t — общее количество документов, поступивших не позднее времени t, $Z_t(d)$ — нормализационное значение:

$$Z_t(d) = \sqrt{\sum_w \left[f(d, w) \cdot \log \frac{N_t}{df_t(w)} \right]^2} \tag{4}$$

.

Теперь можно записать значение похожести двух документов, q и d:

$$\text{sim}_t(d, q) = \sum_w \text{weight}(w, d) \cdot \text{weight}(w, q). \tag{5}$$

Указанные формулы записаны в косинусной метрике, также могут быть использованы метрики Хеллингера, Кульбака–Лейблера и другие.

Для того, чтобы понять, является ли добавленный в момент времени t документ q новым, необходимо вычислить степень его похожести со всеми предыдущими документами. Пусть d^* — документ, максимально похожий на q:

$$d^* = \underset{d}{\text{argmax}}\, \text{sim}_t(d, q). \tag{6}$$

Тогда значение

$$\text{score}_t(q) = 1 - \text{sim}_t(d^*, q) \tag{7}$$

может быть использовано для того, чтобы определить, является ли документ q новым. Новыми будем считать все документы q, у которых значение $\text{score}_t(q)$ больше, чем пороговое значение θ_s. В обратном случае считается что существует документ d^*, достаточно похожий на q и поэтому q не представляет собой сообщение на новую тему. Для того, чтобы определить подходящее значение θ_s, можно использовать размеченный корпус и посчитать значение $\text{sim}_t(d, q)$ среди документов, соответствующих одному и разным событиям.

3.2 Выявление событий с помощью LDA

На следующем примере рассмотрим как тематические модели (topic models) могут быть использованы для задачи распознавания событий. Для этого опишем схему, по которой работает алгоритм, предложенный в [4]. Задача состоит в том, чтобы по данным к фотографиям Flickr[2] распознать все события на определенную тему, проходившие в конкретных городах. Фотографии содержат описания, которые можно считать документами. Основная задача может быть разбита на пять частей:

1. предобработка данных,
2. извлечение города,
3. распознавание темы,
4. распознавание события,
5. оптимизация описания события.

В качестве предобработки, авторы предлагают выполнить следующее: удалить стоп-слова и html теги, провести стемминг слов[3], перевести не английские слова на английский язык используя сервис Google Translate.

Так как задача упоминает отдельные города, необходимо научиться распознавать город по документу. Географические координаты были доступны авторам в 20% фотографий, из этих координат были выявлены города используя сервис Google Tables. Используя технику TF–IDF, в описаниях фотографий с известными городами были извлечены ключевые слова. По этим ключевым словам, появилась возможность назначить фотографиям без геотегов "ближайший"город с точки зрения похожести описаний. Для того чтобы определять похожесть документов, можно воспользоваться подходом, описанным в (3.1). В случаях, когда "ближайший"город выявить не удалось, авторами использовались следующие предположения: считалось что один и тот же автор не мог побывать в один день более чем в двух разных городах, и что путешествие из одного города в другой занимает как минимум два часа. Эти эвристики позволили улучшить классификатор, таким образом более 97% фотографий были привязаны к городу.

Далее необходимо для каждого города кластеризовать документы по темам и рассмотреть только те из них, которые даны в описании задачи. Для распознования тем была использована тематическая модель LDA (Latent Dirichlet Allocation)[5], для определения параметров которой применялось сэмплирование по Гиббсу[6]. LDA работает из предположения, что каждый документ d_i характеризуется случайным распределением над темами, в то время как каждая тема является мультиномиальным распределением над словами.

Оставшаяся часть — извлечение событий и их оптимизация. Для того чтобы алгоритм выявил событие, отвечающее теме k в день D, необходимо чтобы количество документов d_i по этой теме в день D превосходило некоторое пороговое значение θ. Оптимизация событий подразумевает под собой объединение событий на одну тему в последовательные дни и разделение событий в разных городах.

---

[2] http://www.flickr.com

[3] стемминг (stemming) — удаление окончаний у слов для нормализации

Авторы статьи тестировали алгоритм на трех разных вариантах условия задачи, в таблице 1 приведены результаты по каждому из них.

Данные	Точность	Полнота	F-мера
#1	80.98	19.25	31.10
#2	91.21	77.85	84.00
#3	90.76	81.91	86.11

Таблица 1: Точность, полнота и F-мера алгоритма при разных условиях задачи

По результатам можно видеть что в первом случае алгоритм справляется со своей задачей существенно хуже, чем в других. Авторы объясняют это тем, что в задаче #1 необходимо было находить научные конференции, их ключевые слова нельзя ограничить конкретным набором слов, поэтому полнота алгоритма низкая. В задаче #2, #3 напротив удалось обозначить необходимые ключевые слова, о чем свидетельствуют результаты.

3.3 Information nuggets

Рассмотрим подход к извлечению событий, описанный в [1]. Авторы ставят перед собой задачу составить алгоритм описания событий в социальной сети Твиттер. Были использованы данные, полученные во время торнадо Joplin в 2011 году. Данные представляют из себя сообщения пользователей, содержащие хештег #joplin, собранные 22 мая 2011 года на протяжении нескольких часов, пока плотность сообщений не стала относительно низкой. Авторы ставили цель извлекать из потока сообщений так называемые золотые самородки информации — короткие и информативные сообщения, описывающие происходящие события. По этой причине подход назван information nuggets.

Авторы видели основную проблему в том, что даже при наличии большого количества сообщений об одном событии, ими трудно пользоваться, потому что они имеют разную природу. Например это может быть сообщение очевидца о происходящем стихийном бедствии или сообщение о перечислении правительством средств на восстановление разрушенных построек. Статья предлагает делить сообщения на пять разных категорий по степени информативности:

- *Персональное*: информация в сообщении может быть полезна только автору и его кругу общения. Она не является интересной для людей, которые не знают автора сообщения непосредственно.

- *Информативное (напрямую)*: сообщение может быть полезно людям вне круга общения автора, и эта информация написана прямым участником или очевидцем событий.

- *Информативное (косвенно)*: сообщение может быть полезно людям вне круга общения автора, при этом автор пишет о том, что он слышал по телевидению, радио или любому другому источнику информации.

- *Информативное (напрямую или косвенно)*: сообщение может быть полезно людям вне круга общения автора, но невозможно ответить на вопрос как автор связан с происходящими событиями.

- *Другое*: сообщение или не на английском языке, или не может быть классифицировано.

В дальнейшем рассматриваются только сообщения с информативным типом, так как они наиболее вероятно содержат полезные сведения. Затем сообщения разбиваются на подтипы по направленности информации. Всего авторами использовалось 32 разных подтипа, среди которых:

- *Предостережения и советы*: документ содержит предупреждение о возможном опасном происшествии.
- *Жертвы и разрушения*: в тексте сообщается о потерях, вызванных стихийным бедствием.
- *Сбор средств*: сообщения описывают пожертвования денег и предметов пострадавшим от чрезвычайного происшествия.

Для того, чтобы классифицировать сообщения по типам и подтипам, в алгоритме используется наивный байесовский классификатор, который предварительно тренируется на размеченных данных. В классификаторе используется большое количество свойств, бинарных, скалярных и текстовых. В таблице 2 приведены результаты работы алгоритма на некоторых подтипах. Для тестирования использовались размеченные вручную сообщения.

Подтип	Точность	Полнота	F-мера
Предостережения и советы	0.618	0.598	0.605
Жертвы и разрушения	0.578	0.645	0.610
Сбор средств	0.546	0.632	0.585

Таблица 2: Результаты работы алгоритма information nuggets.

4 Теоретическая часть предложенного решения

Эта работа ставит перед собой задачу разработать алгоритм нахождения событий в потоке сообщений пользователей сети Твиттер. В качестве данных были использованы сообщения с 4 июня 2013 года по 31 июня 2013 года, содержащие в себе хэштег

#texas. Всего корпус включает порядка 240 тысяч сообщений и 1.5 миллиона слов. Приведем псевдокод алгоритма, а затем рассмотрим каждый шаг более подробно.

Алгоритм 1: Извлечение событий из сообщений Твиттера

Исходные параметры: множество сообщений Ω, параметры алгоритма M, L, l_D, J, R параметры для HDP модели

Результат: для каждого события набор ключевых слов и частота сообщений по теме события

1 в частотной характеристике сообщений из Ω найти K экстремумов, пусть они соответствуют временам $t_1, t_2, \ldots, t_K$;
2 **для каждого** $t \in \{t_1, t_2, \ldots, t_K\}$ **выполнять**
3 извлечь M ключевых слов из сообщений в Ω в некотором радиусе R момента времени t;
4 сформировать фиктивные документы $D_0 = \left\{D_0^j\right\}_{j=1}^{J}$, каждый из которых имеет длину l_D и состоит из полученных на шаге 3 ключевых слов пропорционально их весу;
5 добавить D_0 к множеству Ω;
6 каждому сообщению из Ω назначить тему, с помощью модифицированной модели HDP (Hierarchical Dirichlet Process);
7 пусть документам D_0 была назначена тема k, удалить D_0 из Ω;
8 найти частотную характеристику сообщений в Ω с темой k;
9 **если** *событие с похожей частотной характеристикой уже было найдено* **тогда**
10 перейти на следующую итерацию цикла;
11 **иначе**
12 найти L ключевых слов из сообщений в Ω с темой k;
13 добавить ключевые слова, частотную характеристику в результат;
14 **конец условия**
15 **конец цикла**

4.1 Нахождение экстремумов

Понятие экстремума в этом разделе отличается от привычного определения в математическом анализе. Назовем экстремумом те точки, в которых плотность сообщений позволяет утверждать что в этот момент произошло событие. В первую очередь, понятие экстремума для задачи выявления событий должно быть локальным. С другой стороны не все локальные максимумы можно корректно обозначить как события. Задача в общем виде получила развитие в контексте обработки сигналов [7]. Существует множество методов, каждый из которых будет больше подходить к определенному типу данных. Алгоритм нахождения экстремумов может меняться вне зависимости от других составных частей исходного алгоритма. Несколько возможных подходов нахождения экстремумов для функции $f(x)$ с сеточной областью определения X перечислены ниже:

- Алгоритм помечает как экстремумы все точки, в которых функция достигает максимума в некотором окне, и при этом ее значение больше некоторого заранее определенного θ. Опционально θ может зависеть от среднего значения функции и других статистических характеристик.

- Для любых двух значений аргумента x и y, где $x < y$ определим функции $T(x, y)$ (travel) и $R(x, y)$ (rise):

$$\begin{aligned} T(x,y) &= \sum_{x \le k < y} |f(k+1) - f(k)|, \\ R(x,y) &= f(y) - f(x) + \epsilon, \end{aligned} \tag{8}$$

  где ϵ — некоторое небольшое значение. Тогда значения $T(x, y)/R(x, y)$ соответствуют пику функции между точками x и y. Алгоритм может находить все экстремумы, где значение T/R больше некоторого порога.

- Подход заключается в том, чтобы выделить в функции стандартную пиковую подфункцию, например функцию Гаусса. Для этого применяются согласованные фильтры. Пусть $g(x)$ — функция Гаусса с некоторыми μ и σ:

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}. \tag{9}$$

  Тогда можно определить степень похожести исходной функции на функцию $g(x)$:

$$\text{sim}(f, g) = \frac{(f, g)}{\|f\| \cdot \|g\|}. \tag{10}$$

  В косинусной метрике выражение примет следующий вид:

$$\text{sim}(f, g) = \frac{\sum\limits_{x \in X} f(x) \cdot g(x)}{\sqrt{\sum\limits_{x \in X} f^2(x)} \cdot \sqrt{\sum\limits_{x \in X} g^2(x)}}. \tag{11}$$

  Аналогично предыдущим случаям, будем считать что функции "похожи"если значение $\text{sim}(f, g)$ не меньше некоторого θ.

- Один из способов найти экстремумы — применить смешанную модель (mixture model) к входной функции. Компонентами модели могут быть нормальные распределения. Затем нужно найти параметры модели используя EM алгоритм или сэмплирование по Гиббсу. Этот метод будет корректно обрабатывать случай, когда экстремумы накладываются друг на друга или расположены очень близко.

Каждый из перечисленных методов можно комбинировать со сглаживающими фильтрами если шум мешает выделить события. Для исходной задачи был выбран первый метод из списка. Данные показали что экстремумы сильно выделяются на фоне средней плотности сообщений и расположены на значительном расстоянии. Таким образом выбранный метод выдает приемлемые результаты.

4.2 Извлечение ключевых слов

Для того, чтобы использовать на шаге 6 модель HDP с частичным обучением, необходимо знать ключевые слова события, которое мы хотим проследить с помощью тематической модели. Их извлечение происходит на шаге 3 и 12. В реализации

алгоритма учитывая особенности сети Твиттер был реализован наиболее простой способ извлечь M ключевых слов из набора сообщений, а именно отсортировать слова по невозрастанию частоты и взять первых M слов.

Этот способ показал хорошие результаты потому что события в сети Твиттер носят "взрывной"характер и частота релевантных слов во время события в десятки и сотни раз превосходит нормальную частоту этих слов. В случае когда "взрывного"характера событий не наблюдается, при извлечении ключевых слов можно использовать технику TF–IDF. Также в этом случае можно рассматривать биграммы и применять различные методы NLP[4] к анализируемому тексту.

4.3 Модифицированная модель HDP

Для извлечения всех сообщений, принадлежащих событию, в алгоритме используется модифицированная модель HDP , в которую добавлена возможность частичного обучения.

4.3.1 Иерархический процесс Дирихле

Пусть $(\Theta, \mathcal{B})$ — измеримое пространство, с вероятностной мерой G_0. Пусть α_0 — положительное действительное число. Определим процесс Дирихле $\mathrm{DP}(\alpha_0, G_0)$ как случайное распределение вероятностной меры G над $(\Theta, \mathcal{B})$, такое, что для любого конечного измеримого разбиения Θ $(A_1, A_2, \ldots, A_r)$, случайный вектор $(G(A_1), G(A_2), \ldots, G(A_r))$ будет распределен согласно конечномерному распределению Дирихле с параметрами $(\alpha_0 G_0(A_1), \alpha_0, G_0(A_2), \ldots, \alpha_0 G_0(A_r))$:

$$(G(A_1), G(A_2), \ldots, G(A_r)) \sim \mathrm{Dir}(\alpha_0 G_0(A_1), \alpha_0, G_0(A_2), \ldots, \alpha_0 G_0(A_r)). \tag{12}$$

Иерархический процесс Дирихле это распределение над множеством вероятностных мер над $(\Theta, \mathcal{B})$. Процесс определяет множество вероятностных мер G_j, по одному на каждую группу, и глобальную меру G_0. Глобальная мера G_0 распределена согласно процессу Дирихле с параметрами γ и базовым распределением H:

$$G_0 \mid \gamma, H \sim \mathrm{DP}(\gamma, H). \tag{13}$$

Групповые меры G_j условно независимы при заданном G_0 и имеют следующее распределение:

$$G_j \mid \alpha_0, G_0 \sim \mathrm{DP}(\alpha_0, G_0). \tag{14}$$

Гиперпараметрами HDP являются базовое распределение H и концентрационные параметры γ и α_0. Иерархический процесс Дирихле может быть использован как априорное распределение над параметрами групповых данных. Для каждого j пусть $\theta_{j1}, \theta_{j2}, \ldots$ — независимые распределенные по закону G_j случайные величины. Каждая величина θ_{ji} будет соответствовать наблюдаемому значению x_{ji}. Правдоподобие может быть записано в следующем виде:

$$\begin{aligned} &\theta_{ji} \mid G_j \sim G_j, \\ &x_{ji} \mid \theta_{ji} \sim F(\theta ji). \end{aligned} \tag{15}$$

Процесс, описанные выше носит название смешанная модель для иерархического процесса Дирихле (hierarchical Dirichlet process mixture model)[8].

[4]Natural Language Processing — обработка текстов на естественном языке

4.3.2 Chinese Restaurant Franchise

HDP — непараметрическая тематическая модель, процесс генерации данных по ней может быть описан в терминах процесса "chinese restaurant franchise"(CRF). В этом процессе существует сеть ресторана, в каждом из которых одинаковое меню. В ресторане находится неограниченное число столов, за каждым столом неограниченное число мест; каждый посетитель, заходя в ресторан, садится либо за уже занятый стол, либо за новый стол. На каждом непустом столе в ресторане заказано одно блюдо из меню, которое едят все посетители, сидящие за этим столом. Блюдо на стол заказывает первый человек, севший за этот стол.

В CRF группам соответствуют рестораны, а параметры θ_{ji} считаются посетителями. Переменные $\phi_1, \ldots, \phi_K$ составляют единое меню для сети ресторанов, эти переменные распределены согласно базовому распределению H. Пусть ψ_{jt} — переменная, связанная со столом t в ресторане i и обозначающее блюдо, которое заказано за этим столом.

Каждый посетителю θ_{ji} соответствует стол ψ_{jt}, в то время как каждому столу ψ_{jt} соответствует одно блюдо ϕ_k. Для того, чтобы указать это соответствие, используются индексы:

- t_{ji} — индекс, связывающий посетителя θ_{ji} со столом ψ_{jt}, за которым сидит этот посетитель.
- k_{jt} — индекс, связывающий стол ψ_{jt} с блюдом ϕ_k, которое заказано для этого стола.

Для описания процесса, необходимо ввести следующие обозначения:

- n_{jtk} — количество посетителей в ресторане j за столом t, едящих блюдо k;
- $n_{jt\cdot}$ — количество посетителей в ресторане j за столом t;
- $n_{j\cdot k}$ — количество посетителей в ресторане j, едящих блюдо k;
- m_{jk} — количество столов в ресторане j, на которых заказано блюдо k;
- $m_{j\cdot}$ — количество столов в ресторане j;
- $m_{\cdot k}$ — количество столов, на которых заказано блюдо k;
- $m_{\cdot\cdot}$ — общее количество столов.

Запишем условное распределение θ_{ji}, где G_j исключены интегрированием:

$$\theta_{ji} \mid \theta_{j1}, \theta_{j2}, \ldots, \theta_{j,i-1}, \alpha_0, G_0 \sim \sum_{i=1}^{m_j} \frac{n_{jt\cdot}}{i - 1 + \alpha_0} \delta_{\psi_{jt}} + \frac{\alpha_0}{i - 1 + \alpha_0} G_0. \tag{16}$$

Аналогично, интегрируя по G_0, получим:

$$\psi_{jt} \mid \psi_{11}, \psi_{12}, \ldots, \psi_{21}, \ldots, \psi_{j,t-1}, \gamma, H \sim \sum_{k=1}^{K} \frac{m_{\cdot k}}{m_{\cdot\cdot} + \gamma} \delta_{\phi_k} + \frac{\gamma}{m_{\cdot\cdot} + \gamma} H. \tag{17}$$

Уравнение (16) описывает выбор стола посетителем, в то время как уравнение (17) определяет выбор блюда.

Если перевести используемые термины в область тематических моделей, посетители будут соответствовать словам, рестораны документам, столы внутренним темам документов, блюда внешними темами всего корпуса[8].

4.3.3 Модификация частичной обучаемости

HDP является моделью без учителя, она не требует размеченных данных для того чтобы назначить документам соответствующие темы. Для того, чтобы контролировать процесс определения тем, необходимо соответствующе изменить модель. В алгоритме 1 необходимо добиться от HDP того, чтобы она выделила ключевые слова из некоторого множества $\mathcal{K}$ в отдельную тему k_0. Есть несколько способов добиться требуемого результата, один из них описан ниже [9].

Составим из ключевых слов в $\mathcal{K}$ документ длины M, в котором количество каждого ключевого слова пропорционально его весу. Перед тем как находить параметры HDP для корпуса документов, добавим J сгенерированных документов в корпус.

Для того, чтобы извлечь параметры из модели HDP в алгоритме применяется сэмплирование по Гиббсу. В этом методе изначальные документы каким-то образом распределяются по темам, а затем итеративно каждое слово удаляется из состояния и назначается на новое место, согласно уравнениям (16) и (17). Краткий псевдокод сэмплирования по Гиббсу для модели HDP приведен ниже:

Алгоритм 2: Сэмплирование по Гиббсу

Исходные параметры: множество документов Ω, параметры для HDP модели α_0, γ

```
1  назначить всем словам тему 0;
2  повторять
3      для каждого документа d ∈ Ω выполнять
4          для каждого слова w ∈ d выполнять
5              удалить w из состояния модели;
6              выбрать внутреннюю тему t согласно уравнению (16);
7              если выбрана новая внутренняя тема t тогда
8                  выбрать внешнюю тему k согласно (17);
9                  назначить внутренней теме t внешнюю тему k;
10             иначе
11                 извлечь назначенную ранее внешнюю тему k для внутренней
                   темы t;
12             конец условия
13             назначить слову w внутреннюю тему t и внешнюю тему k;
14         конец цикла
15     конец цикла
16 до тех пор, пока не прошло L итераций;
```

Для того, чтобы внести в модель частичную обучаемость, необходимо по-разному обрабатывать оригинальные документы и дополнительные. Для первоначальных документов должен сохраниться родной алгоритм, а словам в дополнительном доку-

менте всегда будем назначать внешнюю тему k_0. Приведем псевдокод для модифицированной HDP:

Алгоритм 3: Сэмплирование по Гиббсу с частичным обучением

Исходные параметры: множество документов Ω, параметры для HDP модели α_0, γ, k_0

```
1  назначить всем словам тему 0;
2  повторять
3      для каждого документа d ∈ Ω выполнять
4          для каждого слова w ∈ d выполнять
5              если документ d дополнительный тогда
6                  назначить слову w внутреннюю тему 0 и внешнюю тему k_0;
7              иначе
8                  удалить w из состояния модели;
9                  выбрать внутреннюю тему t согласно уравнению (16);
10                 если выбрана новая внутренняя тема t тогда
11                     выбрать внешнюю тему k согласно (17);
12                     назначить внутренней теме t внешнюю тему k;
13                 иначе
14                     извлечь назначенную ранее внешнюю тему k для внутренней
                       темы t;
15                 конец условия
16                 назначить слову w внутреннюю тему t и внешнюю тему k;
17             конец условия
18         конец цикла
19     конец цикла
20 до тех пор, пока не прошло L итераций;
```

Изменяя параметры M и J можно определять насколько сильной будет привязка к конкретным ключевым словам для искомой темы. Эксперементально установлено что в качестве параметра k_0 можно взять значение 1.

5 Реализация и тестирование решения

Для реализации программы, описанной алгоритмом 1, был выбран язык Java. Выбор ориентирован тем, что в качестве свободной реализации для модели HDP использовалась реализация на языке Java, написанная Arnim Bleier et al. Java объектно-ориентированный язык высокого уровня, что оказалось очень удобным для реализации алгоритма 1.

5.1 Входные данные

Список литературы

[1] Imran, Elbassuoni, Castillo, Diaz and Meier. Extracting Information Nuggets from Disaster-Related Messages in Social Media. 2013.

[2] Xun Wang, Feida Zhu, Jing Jiang, Sujian Li. Real Time Event Detection in Twitter. 2011.

[3] Thorsten Brants, Francine Chen, Ayman Farahat. A System for New Event Detection. 2003.

[4] Konstantinos N. Vavliakis, Fani A. Tzima, and Pericles A. Mitkas. Event Detection via LDA for the MediaEval2012 SED Task. 2012.

[5] David M. Blei, Andrew Y. Ng, Michael I. Jordan. Latent Dirichlet Allocation. 2003.

[6] Tom Griffiths. Gibbs sampling in the generative model of Latent Dirichlet Allocation.

[7] Girish Keshav Palshikar. Simple Algorithms for Peak Detection in Time-Series.

[8] Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, David M. Blei. Hierarchical Dirichlet Processes. 2005.

[9] Ramnath Balasubramanyan, William W. Cohen, Matthew Hurst. Modeling corpora of timestamped documents using semisupervised nonparametric topic models.