***Trainer – основной класс, обучающий модель***

- метод ***TrainBlock*** – работает на нодах

*вход:* документы и старые theta блока в матричном виде

*что делает:* бегает по документам блока, обновляет p(t|d,w) и theta\_d, копит счетчики n\_{wt}, n\_t.

*выход:* новые theta документов блока и информация о посчитанных по блоку счетчиках n\_{wt}, n\_t

- метод ***ProcessSubCollection*** – работает на нодах

*вход:* подколлекция документов (например, путь к папке)

*что делает:* разбивает её на блоки и вызывает *TrainBlock* для каждого блока.

Перед каждым вызовом считывает с диска (или берет из памяти) документы и старые theta блока. После вызова сохраняет новые theta, шлет мастеру счетчики и замеры качества для синхронизации, получает и сохраняет новые фи.

- метод ***TrainModel*** – работает на мастере

*вход:* коллекция документов (например, путь к папке, или имя)

*выход:* профили тем phi, агрегированные замеры качества на обучении

*что делает:* запускает на нодах *ProcessSubCollection* от нужных подколлекций. Обрабатывает запросы нодов: аггрегирует счетчики согласно Aggregate из Trainer-а, аггрегирует замеры качества согласно Aggregate из соответсвующих QualityEstimator-ов. Применяет регуляризацию к фи, как велит RegularizersManager, и отсылает результат ноду.

**При создании объект инициализируется пользователем кучей параметров:**

1. **число тем topicsNumber**
2. **начальные приближения на фи и теты initial**

Это класс, объект создается с параметрами: “randomPreprocess”, parameters | “gamma”, parameters | “semiSupervised”, parameters…

Основные методы – ***GetInitialPhi, GetInitialTheta.***

1. **стратегия E-шага eStepStrategy**

Это класс, объект создается с параметрами: “proportional” | ”sampling”, s | “maximum”, m | ”alternation”, some\_parameters.

Основной метод – ***UpdateCounters****,* который по посчитанному вектору p(t|,d,w) увеличивает нужным образом счетчики в n\_{wt}, n\_t, n\_{td}, n\_d.

1. **регуляризация M-шага**

Класс ***RegularizersManager***, инициализируется пользователем вектором регуляризаторов, которые он хочет использовать, а также стратегией совместной оптимизации параметров регуряризации.

Регуляризаторы – отдельные классы: ***Regularizer*** и его наследники ***StopWordsRegularizer***, ***AntiCorrelationRegularizer***, ....

Основные методы регуляризатора:

***Preprocess*** – собирает и сохраняет статистики по коллекции до обучения тематической модели

***GetAddition*** –возвращает вектор добавок в числитель для всех тем (слов) и его сумму.

1. **схема итерационного алгоритма (standard / online)**

Это класс, инициализируется режимом “standard” | “online” и набором параметров.

*standard*: гарантируется, что объект будет обучаться всегда по одному и тому же блоку документов. Он хранит у себя старые значения счетчиков n\_{wt}, n\_t. Возвращает разности новых и старых значений.

*online*: объект может обучаться на разных блоках, старых n\_{wt} не хранит, возвращает сами счетчики.

Параметры:

**- размер блока blockSize (в штуках)**

**- флаг use\_old\_theta**

Если false, никакие theta не сохраняются, а *TrainBlock* каждый раз получает на вход рандомные распределения.

**- стратегия усреднений счетчиков PhiAggregation:**

Это класс, объект создается с параметрами:

“sum” | “arithmetic”, rho | “exponential”, kappa

Основной метод -- ***Aggregate***(old, new, blockNumber) --

возвращает значение счетчиков result, агрегированное из new и old.

**- критерий останова итераций по документу documentIterations**

Это класс, объект создается с параметрами:

“maximum”, number | “convergence”, threshold | “mix”, number, threshold

Основные методы – reset (сбрасывает счетчик итераций) и finished (возвращает true, если итерации пора закончить согласно критерию).

В случае режима standard параметры заданы по умолчанию как:

use\_old\_theta = false, averaging = sum, blockSize = collectionSize / process\_num,

documentIterationsStop = DocumentIterationsStop(“maximum”, 1).

В слачае режима online какие-нибудь адекватные онлайновые параметры.

**6. внешний критерий останова iterations**

Это класс, объект создается с параметрами:

“maximum”, number | “convergence”, threshold | “mix”, number, threshold,

Основные методы – reset (сбрасывает счетчик итераций) и finished (возвращает true, если итерации пора закончить согласно критерию).

Это проверяет мастер. Наверное, надо добавить возможность нодам проверять, что они успели просмотреть каждый документ minimum раз.

1. **Способ контроля качества на обучении**

Класс ***QualityEstimatorsManager***, инициализируется пользователем вектором измерителей качества, которые он хочет использовать.

Измерители качества – отдельные классы: ***QualityEstimator*** и его наследники ***PerplexityEstimator***, ***SearchMAPEstimator***, ***CoherenceEstimator***, ...

Каждый измеритель имеет функции-обработчики определенных точек алгоритма, что-то типа:

- после расчета распределения p(t|d,w)

- после прохода по документу

- после прохода по блоку на ноде – ***GetBlockQuality*** возвращает результат по блоку, который будет отправлен мастеру

- при получении информации по блоку на мастере

Функция **Aggregate**, описывающая как происходит агрегация результатов, посчитанных на разных нодах для данной метрики качества.

Каждый измеритель качества инициализируется пользователем своими параметрами.