

Доклад про фреймворк для тематического моделирования

Ilya Kozlov
Avanesov Valeriy

7 июля 2014 г.

PLSA

$$L(\Phi, \Theta) = \sum_{d \in D} \sum_{w \in d} n_{dw} \ln \sum_{t \in T} \phi_{wt} \theta_{td} \rightarrow \max_{\Phi, \Theta}$$

RobustPLSA

$$L(\Phi, \Theta, \Pi) = \sum_{d \in D} \sum_{w \in d} n_{dw} \ln \frac{\sum_{t \in T} \phi_{wt} \theta_{td} + \gamma \pi_{dw} + \varepsilon \pi_w}{1 + \gamma + \varepsilon} \rightarrow \max_{\Phi, \Theta, \Pi}$$

Регуляризованный PLSA

$$\mathcal{L}(\Phi, \Theta) = L(\Phi, \Theta) + R(\Phi, \Theta) \rightarrow \max_{\Phi, \Theta}$$

Запустите уже!!!

```
object PresentationQuickStart extends App {  
  def getTokenizedDocuments(): Iterator[Seq[String]] = {...}  
  
  val textualDocuments = getTokenizedDocuments()  
  val (documents, alphabet) = SingleAttributeNumerator(textualDocuments)  
  
  val numberOfTopics = 25  
  val numberOfIteration = 100  
  val random = new Random()  
  val plsa = new PLSABuilder(numberOfTopics, alphabet,  
    documents, random, numberOfIteration).build()  
  
  val trainedModel = plsa.train // the deepest and the darkest magic  
  val phiArrayArray = TopicHelper.copyMatrixToArray(trainedModel.getPhi)  
  
  TrainedModelSerializer.save(trainedModel, "examples/model")  
  TrainedModelSerializer.load("examples/model")  
  
  val fixedPhi = new FixedPhiBuilder(alphabet,  
    documents, numberOfIteration, trainedModel.phi).build()  
  val newTrainedModel = fixedPhi.train  
  
}
```

ЕМ-алгоритм

- Считаются n_{dwt}
- Затем $\phi_{wt} \propto \sum_d n_{dwt}$ и $\theta_{td} \propto \sum_w n_{dwt}$

Почему стоит объединить матрицу счётчиков и матрицу вероятностей

- матрицы счетчиков n_{wt} и n_{dt} и соответствующие матрицы вероятностей Φ и Θ связаны по смыслу
- Матриц вероятностей много, но каждой соответствует своя матрица счётчиков
- Матрица счётчиков бесполезна без своей матрицы вероятностей
- Матрица вероятностей бесполезна без матрицы счётчиков
- Необходимо ограничить возможность изменения матрицы вероятностей, так чтобы она оставалась нормированной.

Огры

Огр - двухголовое создание.

Головы огра это матрица счётчиков и матрица вероятностей
 Φ и Θ - огры

Что огр умеет?

- 1 Хранить матрицу счётчиков и матрицу вероятностей.
- 2 Изменять матрицу счётчиков
- 3 Перемещать значения из матрицы счётчиков в матрицу вероятностей
- 4 Разреживать матрицу вероятностей

Зачем это надо

- 1 Счётчики и вероятности всегда вместе, не перепутаешь
- 2 Матрица вероятностей всегда нормирована на 1.

$+R(\Theta, \Phi)$

- более человекопонятные темы
- частичное обучение
- LDA
- ВСЕ ЧТО ВЫ ХОТИТЕ

ЕМ-алгоритм

$$\theta_{dt} \propto \sum_d n_{dwt} + \theta_{dt} \frac{\partial R}{\partial \theta_{dt}}(\Theta, \Phi)$$

$$\phi_{wt} \propto \sum_d n_{dwt} + \phi_{wt} \frac{\partial R}{\partial \phi_{wt}}(\Theta, \Phi)$$

Интерфейс

```
abstract class Regularizer {  
  def apply(phi: Map[AttributeType, AttributedPhi],  
            theta: Theta): Float  
  
  final def regularizePhi(phi: AttributedPhi,  
                           theta: Theta): Unit = {  
    regularizePhiImmutable(  
      phi, ImmutableTheta.toImmutableTheta(theta))  
  }  
  
  final def regularizeTheta(  
    phi: Map[AttributeType, AttributedPhi],  
    theta: Theta): Unit = {  
    regularizeThetaImmutable(  
      phi.map { case (attr, phi) =>  
        (attr, ImmutablePhi.toImmutablePhi(phi))  
      }, theta)  
  }  
  
  protected def regularizePhiImmutable(phi: AttributedPhi,  
                                         theta: ImmutableTheta): Unit  
  
  protected def regularizeThetaImmutable(  
    phi: Map[AttributeType, ImmutablePhi], theta: Theta): Unit  
}
```

Но зачем?

Есть мнение, что каждый документ и каждое слово относятся не ко всем темам.

```
def apply(matrix: MatrixForSparsifier,  
numberOfIteration: Int): Unit
```

MatrixForSparsifier ?

Единственный изменяющий состояние метод – def
setZero(rowIndex: Int, columnIndex: Int)

Рандомом плохо?

- 1 Да, плохо
- 2 Невыпуклость задачи
- 3 Быстрее сходится, лучше максимум

Варианты

- 1 Рандом
- 2 Равномерно (fixed ϕ)
- 3 Гиббсом

Собираем всю братию вместе

```
val plsa = new PLSABuilder(numberOfTopics,  
    alphabet, documents, random, numberOfIteration)  
    .addRegularizer(new SymmetricDirichlet(0.1f, 0.1f))  
    .setInitialApproximationGenerator(  
        new GibbsInitialApproximationGenerator(random))  
    .setThetaSparsifier(  
        new ThresholdSparsifier(0.01f, 10, Integer.MAX_VALUE))  
    .setPhiSparsifier(new ZeroSparsifier)  
    .build()
```

Было

$$d = (w_1 \dots w_n)$$

$$z_{di} \sim \theta_d. \quad w_{di} \sim \phi_{z_{di}}.$$

$$L(\Phi, \Theta) = \sum_{d \in D} \sum_{w \in d} n_{dw} \ln \sum_{t \in T} \phi_{wt} \theta_{td} \rightarrow \max_{\Phi, \Theta}$$

Стало

$$d = (w_1^1 \dots w_{n_1}^1), (w_1^2 \dots w_{n_2}^2), \dots (w_1^l \dots w_{n_l}^l)$$

$$z_{di^l} \sim \theta_d. \quad w_{di^l}^l \sim \phi_{z_{di^l}}^l.$$

$$L(\Phi_1, \Phi_2, \Theta) = \sum_{(d_1, d_2) \in D} \sum_{w \in d^1} n_{dw} \ln \sum_{t \in T} \phi_{wt}^1 \theta_{td} + \\ \sum_{w \in d^2} n_{dw} \ln \sum_{t \in T} \phi_{wt}^2 \theta_{td} \rightarrow \max_{\Phi, \Theta}$$

Менее формальное определение

Документы и атрибуты

В отличие от обычного тематического моделирования в мультязычной тематической модели каждому документу может соответствовать несколько текстов, например на разных языках.

При этом распределение по темам у всех документов общее.

Вот как это выглядит:

en → "integral, gradient, matrix"

ru → "матрица, градиент, производная, логарифм"

pl → "пшеИнтеграл, пшеМатрица, пшеПеременная "

Как это реализовано?

атрибуты

- Каждому языку ставится в соответствие свой атрибут. (Word("ru"), Word("en")...)
- Каждый текстовый документ создаётся как Map(атрибут → текст)
- Каждому атрибуту ставится в соответствие своя матрица Φ (слова/темы), матрица Θ (документы/темы) общая.
- Отображение слов в порядковый номер и наоборот своё для каждого атрибута, но хранится всё в классе *Alphabet*

Не хочу использовать атрибуты

Если вы используете только один атрибут, то, по умолчанию используется DefaultAttributeType.

Как всё это выглядит смотри в PresentationQuickStart

Лепим многоязыковые документы

```
val td1 = new TextualDocument(Map(  
  Word("ru") -> Seq("утки", "утки", "утки"),  
  Word("en") -> Seq("duck", "duck", "duck", "duck", "duck")))  
  
val td2 = new TextualDocument(Map(  
  Word("ru") -> Seq("утки", "утки", "интеграл"),  
  Word("en") -> Seq("duck", "duck", "duck",  
                    "integral", "gradient")))  
  
val td3 = new TextualDocument(Map(  
  Word("ru") -> Seq("градиент", "производная", "интеграл"),  
  Word("en") -> Seq("integral", "logarithm",  
                    "integral", "integral", "gradient")))  
  
val td4 = new TextualDocument(Map(  
  Word("ru") -> Seq("градиент", "производная",  
                    "интеграл", "логарифм")))  
val textualDocuments = Iterator(td1, td2, td3, td4)
```

```
val (documents, alphabet) = Numerator(textualDocuments)

val numberOfTopics = 2
val numberOfIteration = 100
val random = new Random()
val plsa = new PLSABuilder(numberOfTopics, alphabet,
    documents, random, numberOfIteration).build()

val trainedModel = plsa.train // the deepest and the darkest magic

val phiArrayArray =
    TopicHelper.copyMatrixToArray(trainedModel.getPhi(Word("en")))
val thetaArrayArray =
    TopicHelper.copyMatrixToArray(trainedModel.theta)
```

Спасибо за внимание!