ОБРАБОТКА ЕСТЕСТВЕННЫХ ЯЗЫКОВ: ОТ НАИВНОГО БАЙЕСА К LDA

Сергей Николенко

СПбГУ — Санкт-Петербург

16 мая 2020 г.

Random facts:

- 16 мая 1816 г. — «день рождения индоевропеистики»; этим днём датировано предисловие работы Франца Боппа *Über das Conjugationssystem der Sanskritsprache in Vergleichung mit jenem der griechischen, lateinischen, persischen und germanischen Sprache* («О системе спряжений санскрита в сравнении с таковым в греческом, латинском, персидском, и германском языках»)
- 16 мая 1836 г. Эдгар Аллан По женился на своей 13-летней кузине Вирджинии Клемм
- 16 мая 1896 г. в петербургском саду «Аквариум» состоялся первый в России киносеанс
- 16 мая 1924 г. вышел первый номер журнала «Мурзилка», а 16 мая 1929 г. прошла первая церемония вручения «Оскаров» (тогда они так ещё не назывались)
- 16 мая 1970 г. британский хит-парад на три недели возглавила песня «Back Home» в исполнении сборной Англии по футболу; впрочем, в Мексике англичане проиграли уже в четвертьфинале

Задачи обработки естественного языка

Я вгрызаюсь в тело текста

- Работы, связанные с естественным языком, — это одна из ключевых задач для создания искусственного интеллекта.
- А.А. Марков-старший, 1913: численные эксперименты с текстом «Евгения Онегина», языковые модели через марковские цепи
- Ранний оптимизм: 1950-е, Ноам Хомский, Дартмутский семинар.
- Georgetown-IBM experiment, 1954: «через 3-5 лет машинный перевод будет решён».
- Но оказалось, что всё сложно; первая зима нейронных сетей — это отчасти fail проекта по машинному переводу.
- И до сих пор мы, хотя умеем решать связанные с языком задачи всё лучше и лучше, очень далеки от истинного понимания.

Я вгрызаюсь в тело текста

- Почему сложно? Во многом из-за модели окружающего мира, commonsense reasoning.
- Пример — разрешение *анафоры*:
  - мама вымыла раму, и теперь она блестит;
  - мама вымыла раму, и теперь она устала.
- Это пример хорошо определённой задачи, фактически задачи классификации, для которой легко набрать датасет, но она в нетривиальных случаях всё равно очень, очень сложная.
- Какие ещё бывают задачи в обработке естественного языка?

Задачи NLP

(1) Синтаксические задачи:

- *частеречная разметка* (part-of-speech tagging): разметить в заданном тексте слова по частям речи (существительное, глагол, прилагательное...) и, возможно, по морфологическим признакам (род, падеж...);

Задачи NLP

(1) Синтаксические задачи:

- *частеречная разметка* (part-of-speech tagging);
- *морфологическая сегментация* (morphological segmentation): разделить слова в заданном тексте на *морфемы*, т.е. синтаксические единицы вроде приставок, суффиксов и окончаний; для некоторых языков (например, английского) это не очень актуально, но в русском языке морфологии очень много;

Задачи NLP

(1) Синтаксические задачи:

- *частеречная разметка* (part-of-speech tagging);
- *морфологическая сегментация* (morphological segmentation);
- другой вариант задачи о морфологии отдельных слов — *стемминг* (stemming), в котором требуется выделить основы слов, или *лемматизация* (lemmatization), в которой слово нужно привести к базовой форме (например, форме единственного числа мужского рода);

Задачи NLP

(1) Синтаксические задачи:

- *частеречная разметка* (part-of-speech tagging);
- *морфологическая сегментация* (morphological segmentation);
- *стемминг* (stemming)
- *выделение границ предложения* (sentence boundary disambiguation): разбить заданный текст на предложения; задача непростая даже в русском и английском, а в языках вроде китайского весьма нетривиальной становится даже задача *пословной сегментации* (word segmentation), потому что поток иероглифов без пробелов может делиться на слова по-разному;

Задачи NLP

(1) Синтаксические задачи:

- *частеречная разметка* (part-of-speech tagging);
- *морфологическая сегментация* (morphological segmentation);
- *стемминг* (stemming)
- *выделение границ слов и предложений*;
- *распознавание именованных сущностей* (named entity recognition): найти в тексте собственные имена людей, географических и прочих объектов, разметив их по типам сущностей (люди, места, названия компаний и т.п.);

Задачи NLP

(1) Синтаксические задачи:

- *частеречная разметка* (part-of-speech tagging);
- *морфологическая сегментация* (morphological segmentation);
- *стемминг* (stemming)
- *выделение границ слов и предложений*;
- *распознавание именованных сущностей* (named entity recognition);
- *разрешение смысла слов* (word sense disambiguation): выбрать, какой из омонимов, какой из разных смыслов одного и того же слова используется в данном отрывке текста;

Задачи NLP

(1) Синтаксические задачи:

- *частеречная разметка* (part-of-speech tagging);
- *морфологическая сегментация* (morphological segmentation);
- *стемминг* (stemming)
- *выделение границ слов и предложений*;
- *распознавание именованных сущностей* (named entity recognition);
- *разрешение смысла слов* (word sense disambiguation);
- *синтаксический парсинг* (syntactic parsing): по заданному предложению (и, возможно, его контексту) построить его синтаксическое дерево, прямо по Хомскому;

Задачи NLP

(1) Синтаксические задачи:

- *частеречная разметка* (part-of-speech tagging);
- *морфологическая сегментация* (morphological segmentation);
- *стемминг* (stemming)
- *выделение границ слов и предложений*;
- *распознавание именованных сущностей* (named entity recognition);
- *разрешение смысла слов* (word sense disambiguation);
- *синтаксический парсинг* (syntactic parsing);
- *разрешение кореференций* (coreference resolution): определить, к каким объектам или другим частям текста относятся те или иные слова и обороты; частный случай этой задачи — то самое разрешение анафоры, которое мы обсуждали выше.

Задачи NLP

(1) Синтаксические задачи.

(2) Хорошо определённые семантические задачи:

- *языковые модели* (language models): по заданному отрывку текста предсказать следующее слово или следующий символ; эта задача очень важна, например, для распознавания речи;

Задачи NLP

(1) Синтаксические задачи.

(2) Хорошо определённые семантические задачи:

- *языковые модели* (language models);
- *анализ тональности* (sentiment analysis): определить по тексту его тональность, т.е. позитивное ли отношение несёт этот текст или негативное;

Задачи NLP

(1) Синтаксические задачи.

(2) Хорошо определённые семантические задачи:

- *языковые модели* (language models);
- *анализ тональности* (sentiment analysis);
- *выделение отношений* или *фактов* (relationship extraction, fact extraction): выделить из текста хорошо определённые отношения или факты об упоминающихся там сущностях; например, кто с кем находится в родственных отношениях, в каком году основана упоминающаяся в тексте компания и т.д.;

Задачи NLP

(1) Синтаксические задачи.

(2) Хорошо определённые семантические задачи:

- *языковые модели* (language models);
- *анализ тональности* (sentiment analysis);
- *выделение отношений* или *фактов* (relationship extraction, fact extraction);
- *ответы на вопросы* (question answering): дать ответ на заданный вопрос; в зависимости от постановки это может быть или чистая классификация (выбор из вариантов ответа, как в тесте), или классификация с очень большим числом классов (ответы на фактологические вопросы вроде «кто» или «в каком году»), или даже порождение текста (если отвечать на вопросы нужно в рамках естественного диалога).

Задачи NLP

(1) Синтаксические задачи.

(2) Хорошо определённые семантические задачи.

(3) Хуже определённые семантические задачи:

- собственно *порождение текста* (text generation);

Задачи NLP

(1) Синтаксические задачи.

(2) Хорошо определённые семантические задачи.

(3) Хуже определённые семантические задачи:

- собственно *порождение текста* (text generation);
- *автоматическое реферирование* (automatic summarization): по тексту породить его краткое содержание, abstract, так сказать; это можно рассмотреть как задачу классификации, если просить модель выбрать из текста готовые предложения, лучше всего отражающие смысл всего текста, а можно как задачу порождения, если краткое содержание нужно написать с нуля;

Задачи NLP

(1) Синтаксические задачи.

(2) Хорошо определённые семантические задачи.

(3) Хуже определённые семантические задачи:

- собственно *порождение текста* (text generation);
- *автоматическое реферирование* (automatic summarization);
- *машинный перевод* (machine translation): по тексту на одном языке породить соответствующий текст на другом языке;

Задачи NLP

(1) Синтаксические задачи.

(2) Хорошо определённые семантические задачи.

(3) Хуже определённые семантические задачи:

- собственно *порождение текста* (text generation);
- *автоматическое реферирование* (automatic summarization);
- *машинный перевод* (machine translation);
- *диалоговые модели* (dialog and conversational models): поддержать разговор с человеком.

Поиск дубликатов

- Есть и другие задачи, менее специфичные именно для NLP.
- Например, *поиск дубликатов*: как найти похожие тексты? Классический ответ довольно простой:
  - сначала представим текст как множество (слов? или чего?)
  - выберем для них представление (хешем)
  - найдём похожесть между множествами чисел (как это?)

Кстати, об n-граммах

- n-граммы (шинглы) попадаются в NLP часто. Например, для языковых моделей:
  - предсказываем следующее слово по нескольким предыдущим
  - оцениваем через *перплексию*:

$$PP(S) = p(w_1, \dots, w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{\prod_{i=1}^{N} p(w_i \mid w_1, \dots, w_{i-1})}} = 2^{H(S)}$$

  - надо сглаживать:
    - лапласовское сглаживание;
    - откат (backoff) и интерполяция;
    - Kneser-Ney smoothing:

$$p_{KN}(w_i \mid w_{i-1}) = \frac{\max(c(w_{i-1}w_i) - d)}{c(w_{i-1})} + \lambda(w_{i-1})p_{CONT}(w_i),$$

      где d – константная (!) разница между числом биграмм в training set и validation set, $\lambda(w_{i-1}) = \frac{d|\{w : c(w_{i-1}w) > 0\}|}{\sum_w c(w_{i-1}w)}$ – вес интерполяции, $p_{CONT}(w_i) = \frac{|\{w_{i-1} : c(w_{i-1}w) > 0\}|}{|\{(w_{j-1}, w_j) : c(w_{j-1}w_j) > 0\}|}$.

Кстати, об n-граммах

- А ещё n-граммами разумно расширять набор токенов: Криштиану_Роналду, Евгений_Онегин...
- Но добавлять все пары слов было бы странно и контрпродуктивно.
- Какие нужно добавлять биграммы?..

Кстати, об n-граммах

- А ещё n-граммами разумно расширять набор токенов: Криштиану_Роналду, Евгений_Онегин...
- Но добавлять все пары слов было бы странно и контрпродуктивно.
- Какие нужно добавлять биграммы?..
- ...неожиданные! То есть сильно выбивающиеся из предположения независимости:

$$p(\text{Евгений\_Онегин}) \gg p(\text{Евгений})p(\text{Онегин})$$

Общие замечания

- Часто задачу можно выразить как задачу классификации (*категоризации текстов*) или регрессии.
- В таких случаях можно использовать обычные классификаторы: логистическую регрессию, SVM и т.п.
- Вопрос: что давать им на вход?
- Можно просто считать каждое слово своей размерностью и давать векторы документов как счётчики слов (bag of words).
- Почему это может быть плохо?

Общие замечания

- Одна причина – всё будет сильно зависеть от слов, которые как раз ничего не определяют, и стоп-листами это не решить.
- Вариант – *tf-idf веса*:

$$\mathrm{tf}(t, d) = \frac{n_t}{|d|}, \quad \mathrm{idf}(t, D) = \log \frac{|D|}{|\{d \in D \mid t \in d\}|}.$$

- Обычно результат улучшается, если заменить просто число вхождений на tf-idf веса.
- А сейчас чаще всего используют word embeddings, о них позже...

Наивный байесовский классификатор

Категоризация текстов

- Классическая задача машинного обучения и information retrieval – категоризация текстов.
- Дан набор текстов, разделённый на категории. Нужно обучить модель и потом уметь категоризовать новые тексты.
- Атрибуты $a_1, a_2, \ldots, a_n$ – это слова, v – тема текста (или атрибут вроде «спам / не спам»).
- Bag-of-words model: забываем про порядок слов, составляем словарь. Теперь документ – это вектор, показывающий, сколько раз каждое слово из словаря в нём встречается.

Naive Bayes

- Заметим, что даже это – сильно упрощённый взгляд: для слов ещё довольно-таки важен порядок, в котором они идут...
- Но и это ещё не всё: получается, что $p(a_1, a_2, \dots, a_n | x = v)$ – это вероятность *в точности такого набора слов* в сообщениях на разные темы. Очевидно, такой статистики взять неоткуда.
- Значит, надо дальше делать упрощающие предположения.
- Наивный байесовский классификатор – самая простая такая модель: давайте предположим, что все слова в словаре условно независимы при условии данной категории.

Naive Bayes

- Иначе говоря:

$$p(a_1, a_2, \dots, a_n | x = v) = p(a_1 | x = v) p(a_2 | x = v) \dots p(a_n | x = v).$$

- Итак, наивный байесовский классификатор выбирает v как

$$v_{NB}(a_1, a_2, \dots, a_n) = \arg\max_{v \in V} p(x = v) \prod_{i=1}^{n} p(a_i | x = v).$$

- В парадигме классификации текстов мы предполагаем, что разные слова в тексте на одну и ту же тему появляются независимо друг от друга. Однако, несмотря на такие бредовые предположения, naive Bayes на практике работает очень даже неплохо (и этому есть разумные объяснения).

Многомерная модель

- Но в деталях реализации наивного байесовского классификатора есть тонкости.
- Сейчас мы рассмотрим два разных подхода к naive Bayes, которые дают разные результаты: мультиномиальный (multinomial) и многомерный (multivariate).

Многомерная модель

- В многомерной модели документ – это вектор бинарных атрибутов, показывающих, встретилось ли в документе то или иное слово.
- Когда мы подсчитываем правдоподобие документа, мы перемножаем вероятности того, что встретилось каждое слово из документа и вероятности того, что не встретилось каждое (словарное) слово, которое не встретилось.
- Получается модель многомерных испытаний Бернулли. Наивное предположение в том, что события «встретилось ли слово» предполагаются независимыми.

Многомерная модель

- Математически: пусть $V = \{w_t\}_{t=1}^{|V|}$ – словарь. Тогда документ d_i – это вектор длины $|V|$, состоящий из битов B_{it}; $B_{it} = 1$ iff слово w_t встречается в документе d_i.
- Правдоподобие принадлежности d_i классу c_j:

$$p(d_i \mid c_j) = \prod_{t=1}^{|V|} \left(B_{it} p(w_t \mid c_j) + (1 - B_{it})(1 - p(w_t \mid c_j))\right).$$

- Для обучения такого классификатора нужно обучить вероятности $p(w_t \mid c_j)$.

Многомерная модель

- Обучение – дело нехитрое: пусть дан набор документов $D = \{d_i\}_{i=1}^{|D|}$, которые уже распределены по классам c_j (возможно, даже вероятностно распределены), дан словарь $V = \{w_t\}_{t=1}^{|V|}$, и мы знаем биты B_{it} (знаем документы).
- Тогда можно подсчитать оптимальные оценки вероятностей того, что то или иное слово встречается в том или ином классе (при помощи лапласовой оценки):

$$p(w_t \mid c_j) = \frac{1 + \sum_{i=1}^{|D|} B_{it} p(c_j \mid d_i)}{2 + \sum_{i=1}^{|D|} p(c_j \mid d_i)}.$$

Многомерная модель

- Априорные вероятности классов можно подсчитать как $p(c_j) = \frac{1}{|D|}\sum_{i=1}^{|D|} p(c_j \mid d_i)$.
- Тогда классификация будет происходить как

$$
\begin{aligned}
c &= \arg\max_j p(c_j)p(d_i \mid c_j) = \\
&= \arg\max_j \left(\frac{1}{|D|}\sum_{i=1}^{|D|} p(c_j \mid d_i)\right) \prod_{t=1}^{|V|} \left(B_{it}p(w_t \mid c_j) + (1 - B_{it})(1 - p(w_t \mid c_j))\right) = \\
&= \arg\max_j \left(\log(\sum_{i=1}^{|D|} p(c_j \mid d_i)) + \sum_{t=1}^{|V|} \log\left(B_{it}p(w_t \mid c_j) + (1 - B_{it})(1 - p(w_t \mid c_j))\right)\right).
\end{aligned}
$$

Мультиномиальная модель

- В мультиномиальной модели документ – это последовательность событий. Каждое событие – это случайный выбор одного слова из того самого «bag of words».
- Когда мы подсчитываем правдоподобие документа, мы перемножаем вероятности того, что мы достали из мешка те самые слова, которые встретились в документе. Наивное предположение в том, что мы достаём из мешка разные слова независимо друг от друга.
- Получается мультиномиальная генеративная модель, которая учитывает количество повторений каждого слова, но не учитывает, каких слов *нет* в документе.

Мультиномиальная модель

- Математически: пусть $V = \{w_t\}_{t=1}^{|V|}$ – словарь. Тогда документ d_i – это вектор длины $|d_i|$, состоящий из слов, каждое из которых «вынуто» из словаря с вероятностью $p(w_t \mid c_j)$.
- Правдоподобие принадлежности d_i классу c_j:

$$p(d_i \mid c_j) = p(|d_i|)|d_i|! \prod_{t=1}^{|V|} \frac{1}{N_{it}!} p(w_t \mid c_j)^{N_{it}},$$

  где N_{it} – количество вхождений w_t в d_i.
- Для обучения такого классификатора тоже нужно обучить вероятности $p(w_t \mid c_j)$.

Мультиномиальная модель

- Обучение: пусть дан набор документов $D = \{d_i\}_{i=1}^{|D|}$, которые уже распределены по классам c_j (возможно, даже вероятностно распределены), дан словарь $V = \{w_t\}_{t=1}^{|V|}$, и мы знаем вхождения N_{it}.
- Тогда можно подсчитать апостериорные оценки вероятностей того, что то или иное слово встречается в том или ином классе (не забываем сглаживание – правило Лапласа):

$$p(w_t \mid c_j) = \frac{1 + \sum_{i=1}^{|D|} N_{it} p(c_j \mid d_i)}{|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N_{is} p(c_j \mid d_i)}.$$

Мультиномиальная модель

- Априорные вероятности классов можно подсчитать как $p(c_j) = \frac{1}{|D|}\sum_{i=1}^{|D|} p(c_j \mid d_i)$.
- Тогда классификация будет происходить как

$$
\begin{aligned}
c &= \arg\max_j p(c_j)p(d_i \mid c_j) = \\
&= \arg\max_j \left(\frac{1}{|D|}\sum_{i=1}^{|D|} p(c_j \mid d_i) \right) p(|d_i|)|d_i|! \prod_{t=1}^{|V|} \frac{1}{N_{it}!} p(w_t \mid c_j)^{N_{it}} = \\
&= \arg\max_j \left(\log \left(\sum_{i=1}^{|D|} p(c_j \mid d_i) \right) + \sum_{t=1}^{|V|} N_{it} \log p(w_t \mid c_j) \right).
\end{aligned}
$$

Как можно обобщить наивный байес

- В наивном байесе есть два сильно упрощающих дело предположения:
  - мы знаем метки тем всех документов;
  - у каждого документа только одна тема.
- Мы сейчас уберём оба эти ограничения.
- Во-первых, что можно сделать, если мы не знаем метки тем, т.е. если датасет неразмеченный?

Кластеризация

- Тогда это превращается в задачу кластеризации.
- Её можно решать EM-алгоритмом (Expectation-Maximization, используется в ситуациях, когда есть много скрытых переменных, причём если бы мы их знали, модель стала бы простой):
  - на E-шаге считаем ожидания того, какой документ какой теме принадлежит;
  - на M-шаге пересчитываем наивным байесом вероятности $p(w \mid t)$ при фиксированных метках.
- Это простое обобщение.

Тематическое моделирование

Как ещё можно обобщить наивный байес

- А ещё в наивном байесе у документа только одна тема.
- Но это же не так! На самом деле документ говорит о многих темах (но не слишком многих).
- Давайте попробуем это учесть.

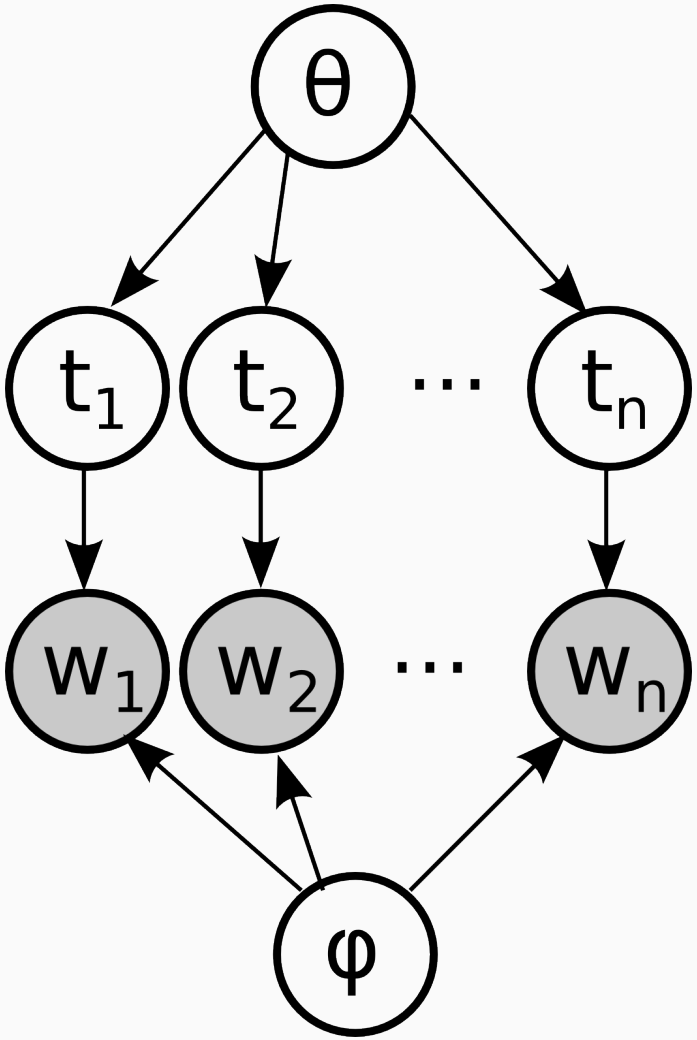
pLSA

- Рассмотрим такую модель:
  - каждое слово в документе d порождается некоторой темой $t \in T$;
  - документ порождается некоторым распределением на темах $p(t \mid d)$;
  - слово порождается именно темой, а не документом: $p(w \mid d, t) = p(w \mid d)$;
  - итого получается такая функция правдоподобия:

$$p(w \mid d) = \sum_{t \in T} p(w \mid t) p(t \mid d).$$

- Эта модель называется probabilistic latent semantic analysis, pLSA (Hoffmann, 1999).

pLSA: графическая модель документа



Пример

- Получается как-то так:

Алгоритм 2. Рациональный EM-алгоритм для тематической модели (2).

Вход: коллекция D, число тем $|T|$, начальные приближения матриц Φ и Θ;
Выход: параметры модели Φ и Θ;

повторять
  обнулить n_{wt}, n_{td}, n_t для всех $d \in D,\ w \in W,\ t \in T$;
  для всех $d \in D,\ w \in d$
    $n_{tdw} := n_{dw}\varphi_{wt}\theta_{td} \,/\, \sum_\tau \varphi_{w\tau}\theta_{\tau d}$ для всех $t \in T$;
    увеличить n_{wt}, n_{td}, n_t на n_{tdw} для всех $t \in T$;
  $\varphi_{wt} := n_{wt}/n_t$ для всех $w \in W,\ t \in T$;
  $\theta_{td} := n_{td}/n_d$ для всех $d \in D,\ t \in T$;
пока Φ и Θ не сойдутся;

pLSA

- Как её обучать? Мы можем оценить $p(w \mid d) = \frac{n_{wd}}{n_d}$, а нужно найти:
  - $\phi_{wt} = p(w \mid t)$;
  - $\theta_{td} = p(t \mid d)$.
- Максимизируем правдоподобие

$$p(D) = \prod_{d \in D} \prod_{w \in d} p(d, w)^{n_{dw}} = \prod_{d \in D} \prod_{w \in d} \left[\sum_{t \in T} p(w \mid t) p(t \mid d) \right]^{n_{dw}}.$$

- Как максимизировать такие правдоподобия?

pLSA

- EM-алгоритмом. На E-шаге ищем, сколько слов w в документе d из темы t:

$$n_{dwt} = n_{dw} p(t \mid d, w) = n_{dw} \frac{\phi_{wt}\theta_{td}}{\sum_{s \in T} \phi_{ws}\theta_{sd}}.$$

- А на M-шаге пересчитываем параметры модели:

$$n_{wt} = \sum_d n_{dwt}, \qquad n_t = \sum_w n_{wt}, \quad \phi_{wt} = \frac{n_{wt}}{n_t},$$
$$n_{td} = \sum_{w \in d} n_{dwt}, \quad \theta_{td} = \frac{n_{td}}{n_d}.$$

- Вот и весь вывод в pLSA.

- Можно даже не хранить всю матрицу n_{dwt}, а двигаться по документам, каждый раз добавляя n_{dwt} сразу к счётчикам n_{wt}, n_{td}.

Алгоритм 2. Рациональный EM-алгоритм для тематической модели (2).

Вход: коллекция D, число тем $|T|$, начальные приближения матриц Φ и Θ;
Выход: параметры модели Φ и Θ;

повторять
  обнулить n_{wt}, n_{td}, n_t для всех $d \in D,\ w \in W,\ t \in T$;
  для всех $d \in D,\ w \in d$
    $n_{tdw} := n_{dw}\varphi_{wt}\theta_{td} \,/\, \sum_\tau \varphi_{w\tau}\theta_{\tau d}$ для всех $t \in T$;
    увеличить n_{wt}, n_{td}, n_t на n_{tdw} для всех $t \in T$;
  $\varphi_{wt} := n_{wt}/n_t$ для всех $w \in W,\ t \in T$;
  $\theta_{td} := n_{td}/n_d$ для всех $d \in D,\ t \in T$;
пока Φ и Θ не сойдутся;

pLSA

- Чего тут не хватает?
  - Во-первых, разложение такое, конечно, будет сильно не единственным.
  - Во-вторых, параметров очень много, явно будет оверфиттинг, если корпус не на порядки больше числа тем.
  - А совсем хорошо было бы получать не просто устойчивое решение, а обладающее какими-нибудь заданными хорошими свойствами.
- Всё это мы можем решить как?

- Правильно, регуляризацией. Есть целая наука о разных регуляризаторах для pLSA (К.В. Воронцов).
- В общем виде так: добавим регуляризаторы R_i в логарифм правдоподобия:

$$\sum_{d \in D} \sum_{w \in d} n_{dw} \ln \sum_{t \in T} \phi_{wt} \theta_{td} + \sum_i \tau_i R_i(\Phi, \Theta).$$

- Тогда в EM-алгоритме на M-шаге появятся частные производные R:

$$n_{wt} = \left[\sum_{d \in D} n_{dwt} + \phi_{wt}\frac{\partial R}{\partial \phi_{wt}}\right]_+,$$

$$n_{td} = \left[\sum_{w \in d} n_{dwt} + \theta_{td}\frac{\partial R}{\partial \theta_{td}}\right]_+$$

- Чтобы доказать, EM надо рассмотреть как решение задачи оптимизации через условия Каруша-Куна-Такера.

pLSA

- И теперь мы можем кучу разных регуляризаторов вставить в эту модель:
  - регуляризатор сглаживания (позже, это примерно как LDA);
  - регуляризатор разреживания: максимизируем KL-расстояние между распределениями ϕ_{wt} и θ_{td} и равномерным распределением;
  - регуляризатор контрастирования: минимизируем ковариации между векторами ϕ_t, чтобы в каждой теме выделилось своё лексическое ядро (характерные слова);
  - регуляризатор когерентности: будем награждать за слова, которые в документах стоят ближе друг к другу;
  - и так далее, много всего можно придумать.

Спасибо!

Спасибо за внимание!