

ОБРАБОТКА ЕСТЕСТВЕННЫХ ЯЗЫКОВ

Сергей Николенко

TRA Robotics — Санкт-Петербург

30 августа 2018 г.

Random facts:

- 30 августа 1590 г. основатель династии Токугава Иэясу вошёл в замок Эдо
- 30 августа 1703 г. произошло первое наводнение в истории Петербурга, вода поднялась более чем на два метра
- 30 августа 1877 г. Эмиль Рено запатентовал праксиноскоп, прародитель мультипликации, а 30 августа 1901 г. Хьюберт Сесил Бут запатентовал электрический пылесос
- 30 августа 2001 г. суд в Сан-Хосе официально выдвинул обвинение против Дмитрия Склярова и компании «Элкомсофт», обвиняемых в снятии защиты с продуктов фирмы «Adobe»
- 30 августа 2002 г. Генри Купрашвили за 3 часа 15 минут переплыл Дарданеллы (12 км) в древнегрузинском стиле, связанный в четырёх местах по рукам и ногам

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.

- 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?

(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...);

(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;

(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);

(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;

(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.);

(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;

(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;

(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.

- (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;

- (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;

(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.;

- (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).

- (1) Синтаксические задачи.
- (2) Хорошо определённые семантические задачи.
- (3) Хуже определённые семантические задачи:
 - собственно *порождение текста* (text generation);

- (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;

- (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;

- (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.

- 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 похожость между множествами чисел (как это?)

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

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

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

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

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, $p_{CONT}(w_i) = \frac{|\{w_{i-1}:c(w_{i-1}w)>0\}|}{|\{(w_{j-1},w_j):c(w_{j-1}w_j)>0\}|}$.

- 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?..

- 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:

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

- 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?

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

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

- 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

- 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, \dots, 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.

- 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.

- 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).

- 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).

- 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.

- 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 :

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

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

- 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):

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

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

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

- 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.

- 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 $p(w_t | c_j)$.
- Правдоподобие принадлежности d_i классу c_j :

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

где N_{it} – количество вхождений w_t в d_i .

- Для обучения такого классификатора тоже нужно обучить вероятности $p(w_t | c_j)$.

- 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):

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

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

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

- 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?

- 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 $p(w | t)$ при фиксированных метках.
- Это простое обобщение.

Спасибо за внимание!