ЛЕКЦИЯ 13. ЯЗЫКОВЫЕ МОДЕЛИ

СЕРГЕЙ НИКОЛЕНКО

1. Введение

Зачастую мы можем понять разговорную речь даже если говорящий произносит слова не очень разборчиво. А почему? Потому что мы "настроены" на язык, на котором говорит оратор. То есть, в каждый момент времени, слущающий ожидает, что говорящий скажет следующую фонему, букву и/или слово из достаточно ограниченного набора, характерного для данного языка, контекста и темы разговора.

Например, в славянских языках нас бы сильно удивила последовательность из пяти согласных подряд, в японском или корейском нас бы удивили звуки «х» и «л», для фино-угорских языков характерны последовательности гласных, в некоторых языках – таких как литовский и эсперанто – весьма ограниченны наборы окончаний различных частей речи.

То есть, говоря о конкретном языке, можно говорить о некоторой формальной грамматике, заданной на фонемах, буквах и словах этого языка.

2. Формальные грамматики

Формальная грамматика (далее просто *«грамматика»*) в теории формальных языков, предложенной Хомским (Noam Chomsky) в начале 60-х — способ описания формального языка, то есть выделения некоторого подмножества из множества всех слов некоторого конечного алфавита.

Грамматика представляет собой четверку – *терминалы* (обозначается Σ), *нетерминалы* (обозначается N) и *набор правил* вида $\alpha \to \beta$ (обозначается R), где одно правило является *начальным* (обозначается S, кроме того, S – это ещё и символ в левой части этого правила в R, т.е. для него $\alpha = S$).

Терминал – объект, непосредственно присутствующий в словах языка, соответствующего грамматике, и имеющий конкретное, неизменяемое значение (обобщение понятия «буквы»).

Нетерминал – объект, обозначающий какую-либо сущность языка (например: формула, арифметическое выражение, команда) и не имеющий конкретного символьного значения. Таким образом, первый встреченный нами нетерминал – S (начальный символ).

По иерархии Хомского грамматики делятся на следующие типы:

- Тип 0 – неограниченные грамматики (unrestricted grammars), где возможны любые правила. Такие грамматики разпознаются *машинами Тьюринга*.
- Тип 1 – контекстно-зависимые грамматики (context-sensitive grammars), с правилами вида

Законспектировал Ян Малаховски.

  - ⋆ $xAy \to x\gamma y$
  - ⋆ $A \to w$

  где, A – нетерминал, w – терминал, а x, y и γ – некоторые строки, состоящие из терминалов и нетерминалов (контекст), где x и y могут быть пустыми строками, а γ – нет.
- Тип 2 – контекстно-свободные грамматики (context-free grammars, CFG, КС-грамматики), с правилами вида
  - ⋆ $A \to BC$
  - ⋆ $A \to w$

  где, A, B, C – нетерминалы, w – терминал. Этот тип грамматик распознается *автоматами с магазинной памятью*. Также следует упомянуть, что любая грамматика с правилами вида $\alpha \to \beta$, где α – нетерминал, может быть переписана в вышеприведенном виде (называемом нормальной формой).
- Тип 3 – регулярные грамматики (regular grammars), с правилами вида
  - ⋆ $A \to wB$
  - ⋆ $A \to w$

  где, A, B– нетерминалы, w – терминал. Этот тип грамматик распознается *конечными автоматами*.

Для практического применения найбольший интерес представляют контекстно-свободные грамматики, поскольку класс языков, распознаваемых ими, достаточно широк, и, в то же время, для них существуют алгоритмы построения *автоматов с магазинной памятью*, распознающих какой-либо язык, по набору правил R соответствующей грамматики. Примеры таких алгоритмов: SLR, LL(1), LALR, ... (см. [1]).

Пример 1

Рассмотрим, например, предложение “люблю грозу в начале мая” и следующую контекстно-свободную грамматику:

- S → сказуемое дополнение
- сказуемое → глагол
- дополнение → существительное определение
- определение → предлог дополнение | существительное

где S – начальный символ грамматики.

Тогда дерево разбора такого предложения будет выглядеть следующим образом:

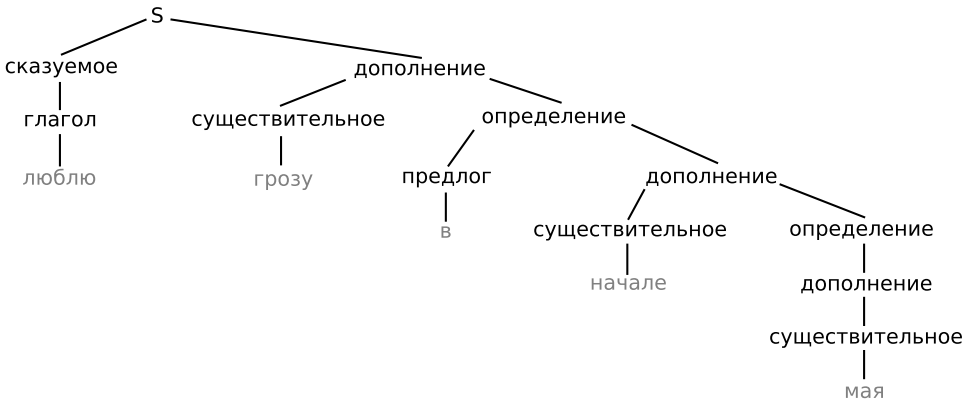

Рис. 1. Дерево разбора предложения «люблю грозу в начале мая». Серым цветом выделены терминалы.

3. Естественные языки

Не смотря на то, что существуют человеческие (те, на которых люди говорят) языки, для которых вполне можно было бы построить приемлимую контекстно-свободную грамматику (так называемые “плановые” языки – например, Esperanto и Ido), большая часть натуральных языков не может похвастаться такой возможностью. Это связано с тем, что в натуральных языках существует неимоверное количество двусмысленных конструкций (как правило ещё и зависимых от контекста), которые невозможно представить КС-грамматикой. Кроме того, существуют фразы, которые вообще являются неправильными с точки зрения грамматики, в то же время используемые нами.

Выхода из данной ситуации два:

- сделать распознаваемый язык очень простым,
- сделать грамматику с огромным количеством правил.

Для машинной обработки второй подход, очевидно, не имеет смысла из-за своей потенциальной трудоёмкости (в идеальном случае нам бы хотелось получать результаты в реальном времени, оратор нас ждать не будет). Первый же подход широко используется.

Итак, мы решили, что для разбора естественных языков мы выберем некоторое ограниченное подмножество грамматических конструкций нашего языка, построим небольшую КС-грамматику и будем ей пользоваться.

4. SCFG

Для прозорливого читателя после предыдущей фразы должно остаться непонятным то, как именно мы будем использовать построенную грамматику. Ведь нас мало интересуют ответы вида «Да/Нет» (правда ли, что вход не противоречит грамматике?) на предложенную парсеру строку символов. В самом начале лекции мы говорили, что после получения очередного элемента из Σ

(терминалы, они же элементы языка) из потока, мы хотим на выходе нашего алгоритма получить список наиболее вероятных вариантов разбора (может быть корректнее "продолжения"? ведь набор терминалов ограничен и мы можем пытаться предсказывать следующий) фразы.

Эту задачу решают *вероятностные КС-грамматики* (stochastic context-free grammars, SCFG, probabilistic context-free grammars, PCFG, ВКС-грамматики). SCFG – это КС-грамматика, где к каждому правилу приписана вероятность его применения, т.е. если КС-грамматика была четвёркой (Σ, N, R, S), то ВКС-грамматика – это пятёрка (Σ, N, R, S, P), где P – отображение из R в промежуток $[0, 1]$ (вероятность правила). В широком смысле, ВКС-грамматики – это такое же расширение КС-грамматик, как *скрытые марковские модели* (см. лекцию #9) – расширение регулярных грамматик.

Пусть все правила грамматики записаны в нормальной форме:

- $A_i \to A_m A_n$
- $A_i \to w_k$

где A_i – нетерминалы, w_k – терминалы, а вероятность каждого правила вычисляется по формуле:

$$p(A_i \to \alpha) = \frac{\#(A_i \to \alpha)}{\sum_\beta \#(A_i \to \beta)}$$

где $\#(A_i \to \alpha)$ – количество "срабатываний" правила $A_i \to \alpha$ для некоторого текста.

Интересующий нас результат – это вероятность $p(S \Rightarrow W = w_1 w_2 ... w_T | G)$, где G – грамматика, S – ее начальный символ, W – некоторая цепочка терминалов w_i (количество которых T), а символ "$\Rightarrow$" следует читать как «порождает».

Тогда с учетом вышеприведенных правил мы могли бы вычислить искомую вероятность для конкретного дерева разбора следующим образом:

$$p(S \Rightarrow W = w_1 w_2 ... w_T | G) = p(S \to A_m A_n) p(A_m \to ...) p(A_n \to ...) ...$$

5. Inside-Outside

Однако, нас интересует $p(S \Rightarrow W)$ в общем случае. Поэтому вводят функции *inside* и *outside*:

$$\begin{aligned} inside(j, A_i, k) &= p(A_i \Rightarrow w_j ... w_k), \text{ определена для } \forall j \le k \\ outside(s, A_i, t) &= p(S \Rightarrow w_1 .. w_{s-1} A_i w_{t+1} ... w_T), \text{ определена для } \forall s \le t \end{aligned}$$

Учитывая, что правила грамматики не зависимы и записаны в нормальной форме, имеем:

$$p(S \Rightarrow W) = \sum_i inside(j, A_i, k) outside(j, A_i, k) = inside(1, S, T)$$

Для самих же функций *inside* и *outside* вводят следующие реккурентные соотношения:

$$\begin{aligned} inside(j, A_i, k) &= \sum_{m,n,l} inside(j, A_m, l) p(A_i \to A_m A_n) inside(l+1, A_n, k) \\ inside(j, A_i, j) &= p(A_i \to w_i) \end{aligned}$$

и

$$\begin{aligned} outside(j, A_i, k) &= \sum_{m,n}[\sum_l p(A_m \to A_n A_i) outside(l, A_m, k) inside(l, A_n, j-1) + \\ &\quad + \sum_l p(A_m \to A_i A_n) outside(j, A_m, l) inside(k+1, A_n, l)] \\ outside(1, S, T) &= 1 \end{aligned}$$

где $l \in [j, k]$, а m, n – пробегают по всем возможным значениям.

6. Адаптация алгоритма Баума-Велха

Теперь мы умеем считать $p(S \Rightarrow W)$, однако парсить книги для получения вероятностей правил из R – достаточно трудоемкое занятие. Поэтому обычно для этих целей используют адаптацию алгоритма Баума-Велха, где функция ξ имеет вид:

$$\xi(i, m, n, s, t) = p(A_i \Rightarrow w_s ... w_t, A_i \to A_m A_n) = \frac{1}{p(S \Rightarrow W)} \times$$
$$\times \sum_k p(A_i \to A_m A_n) inside(s, A_m, k) inside(k+1, A_n, t) outside(s, A_i, t)$$

а пересчет значений вероятностей правил производят по формуле:

$$p(A_i \to A_m A_n) = \frac{\sum_{s,t} \xi(i, m, n, s, t)}{\sum_{m,n,s,t} \xi(i, m, n, s, t)}$$

В качестве начальных распределений обычно используют вероятности веток начального правила S.

7. N-граммы

Другой очень простой и часто используемый метод, применяемый для создания языковых моделей – N-граммы. Пусть известны вероятности

$$p(w_k | w_{k-N}, w_{k-N+1} ... w_{k-1}),$$

т.е. вероятности того, что текущее (при разборе цепочки) слово – это w_k, а N предыдущими словами были $w_{k-N}, w_{k-N+1} ... w_{k-1}$.

Тогда вероятность некоторой цепочки $beg\ w_1 ... w_T\ end$, где beg и end – специальные символы начала и конца строки (цепочки), можно вычислить как $p(beg\ w_1 ... w_T\ end) = p(w_1|beg) p(w_2|w_1) ... p(w_T|w_1 ... w_{T-1}) p(end|w_1 ... w_T)$

Пример 2

Рассмотрим следующий текст:
«однажды ежи прочел книгу
ежи взял и прочел книгу
ежи прочел книгу о себе»

Тогда для N-граммы с $N = 1$ вероятности для слова «ежи» будут иметь вид:

- p(ежи|beg) = 2/3,
- p(ежи|однажды) = 1/3.

А для слова «книгу»: p(книгу|прочел) = 1.

8. *

Список литературы

[1] Альфред В. Ахо, Моника С. Лам, Рави Сети, Джеффри Д. Ульман «Компиляторы: принципы, технологии и инструментарий»; «Compilers: Principles, Techniques, and Tools» — 2 изд. — М.: «Вильямс», 2008. — ISBN 978-5-8459-1349-4