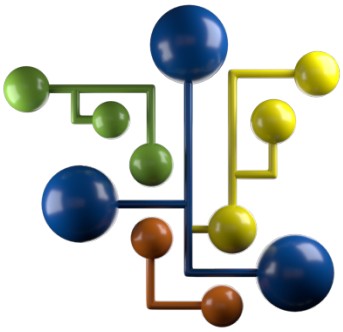

Data Science Academy

www.datascienceacademy.com.br

Introdução à Inteligência Artificial

Análise Sintática

A análise sintática é o processo de analisar uma cadeia de palavras para descobrir a sua estrutura frasal, de acordo com as regras de uma gramática. A tabela abaixo mostra que podemos começar com o símbolo S e pesquisar de cima para baixo em uma árvore que tem as palavras como suas folhas ou podemos começar com as palavras e buscar de baixo para cima de uma árvore que culmina em um S. Contudo, tanto a análise de cima para baixo como a de baixo para cima podem ser ineficientes porque podem acabar repetindo esforços em áreas de espaço de busca que levam a becos sem saída.

Lista de itens	*Regra*
S	
SN SV	*S* → *NP SV*
SN SV Adjetivo	*SV* → *SV Adjetivo*
SN Verbo Adjetivo	*SV* → *Verbo*
SN Verbo **morto**	*Adjetivo* → **morto**
SN **está morto**	*verbo* → **está**
Artigo Substantivo **está morto**	*SN* → *Artigo Substantivo*
Artigo **wumpus está morto**	*Substantivo* → **wumpus**
o wumpus está morto	*Artigo* → **o**

Considere as duas sentenças a seguir:

Have the students in section 2 of Computer Science 101 take the exam.
Have the students in section 2 of Computer Science 101 taken the exam?

Embora compartilhem as 10 primeiras palavras, essas sentenças têm análises sintáticas muito diferentes porque a primeira é um comando e a segunda é uma pergunta. Um algoritmo de análise da esquerda para a direita teria de pressupor que a primeira palavra é parte de um comando ou de uma pergunta, e não seria capaz de saber se a suposição é correta até pelo menos a décima primeira palavra, take ou taken. Se fizer a suposição errada, o algoritmo terá de percorrer de volta toda a distância até a primeira palavra e analisar novamente toda a sentença sob outra interpretação.

Para evitar essa fonte de ineficiência, podemos usar programação dinâmica: cada vez que analisamos uma subcadeia, armazenamos os resultados de modo que não tenhamos que reanalizálos mais tarde. Por exemplo, uma vez que descobrimos que “os alunos na seção 2 de Ciência da Computação 101” é um SN, podemos registrar o resultado em uma estrutura de dados conhecida como diagrama. Os algoritmos que fazem isso são chamados de **analisadores de diagrama**. Por estarmos tratando com gramática livre de contexto, qualquer sintagma que for encontrado no contexto de uma ramificação do espaço de busca pode funcionar muito bem em qualquer outra ramificação de espaço de busca. Existem muitos tipos de analisadores de

diagrama; descreveremos uma versão de baixo para cima chamada de algoritmo CYK, que tem por trás seus inventores, John Cocke, Daniel Younger e Tadeo Kasami.

O algoritmo CYK é mostrado na figura abaixo. Note que ele exige uma gramática com todas as regras em um de dois formatos muito específicos: regras lexicais da forma X → palavra e regras sintáticas da forma X → Y Z. Esse formato da gramática, chamado de forma normal de Chomsky, pode parecer restritivo mas não é: qualquer gramática livre de contexto pode ser transformada automaticamente na forma normal de Chomsky.

```
função ANÁLISE-CYK (palavras, gramática) retorna P, uma tabela de probabilidades
  N ← COMPRIMENTO (palavras)
  M ← o número de símbolos não terminais na gramática
  P ← uma matriz de tamanho [M, N, N], inicialmente todos 0
  /* Inserir regras lexicais para cada palavra */
  para i = 1 até N faça
    para cada regra da forma (X → palavras_i [p]) faça
      P[X, i, 1] ← p
  /* Combine a primeira e a segunda partes do lado direito das regras, da curta para a longa */
  para comprimento = 2 até N faça
    para início = 1 até N − comprimento + 1 faça
      para comp1 = 1 até N − 1 faça
        comp2 ← comprimento − comp1
        para cada regra da forma (X → Y Z [p]) faça
                P[X, início, comprimento] ← MAX(P[X, início, comprimento],
                P[Y, início, comp1] × P[Z, início + comp1, comp2] × p)
  retornar P
```

O algoritmo CYK utiliza o espaço de O(n2m) da tabela P, onde n é o número de palavras na sentença e m é o número de símbolos não terminais na gramática, que leva o tempo O(n3m). (Uma vez que m é constante para uma gramática particular, é comumente descrito como O(n3).) Nenhum algoritmo pode fazer melhor para gramáticas livres de contexto em geral, embora haja algoritmos mais rápidos em gramáticas mais restritas. Na verdade, é quase um truque para que o algoritmo complete no tempo O(n3), dado que é possível que uma sentença tenha um número exponencial de árvores de análise sintática. Considere a sentence:

Fall leaves fall and spring leaves spring

É ambíguo porque cada palavra (exceto “and”) pode ser um substantivo ou um verbo, e “fall”e “spring” podem ser também adjetivos. (Por exemplo, um significado de “Fall leaves fall” é equivalente a “Outono abandona outono.) A sentença tem quatro análises:

[*S* [*S* [*SN* Fall leaves] fall] e [*S* [*SN* spring leaves]spring]]
[*S* [*S* [*SN* Fall leaves] fall] e [*S* spring [*SV* leaves spring]]
[*S* [*S* Fall [*SV* leaves fall]] e [*S* [*SN* spring leaves] spring]
[*S* [*S* Fall [*SV* leaves fall]] e [*S* spring [*SV* leaves spring]].

Se tivéssemos subsentenças conjuntas c ambíguas de duas maneiras, teríamos 2c formas de escolher análises para as subsentenças. Como o algoritmo CYK processa essas árvores de análise sintática de 2c no tempo O(c3)? A resposta é que ele não examina todas as árvores de análise; tudo o que faz é o cálculo da probabilidade da árvore mais provável. As subárvores estão todas representadas na tabela P e, com um pouco de trabalho, poderíamos enumerar todas (em tempo exponencial), mas a beleza do algoritmo CYK é que não temos que enumerá las, a menos que queiramos.

Na prática, geralmente não estamos interessados em todas as análises, apenas nas melhores ou nas poucas melhores. Pense no algoritmo CYK como a definição do espaço de estados completo definido pelo operador “aplicar regra de gramática”. É possível pesquisar apenas uma parte desse espaço usando a busca A*. Cada estado nesse espaço é uma lista de itens (palavras ou categorias), como mostrado na tabela analítica da base para o topo (tabela acima). O estado inicial é uma lista de palavras, e um estado objetivo é o único item S. O custo de um estado é o inverso de sua probabilidade, tal como definido pelas regras aplicadas até agora, e há várias heurísticas para estimar a distância que falta para a meta, a melhor heurística vindo de aprendizagem de máquina aplicada a um corpus de sentenças. Com o algoritmo A* não temos de buscar o espaço de estado inteiro, e temos a garantia de que a primeira análise encontrada será a mais provável.

Referências:

Livro: Inteligência Artificial
Autor: Peter Norvig