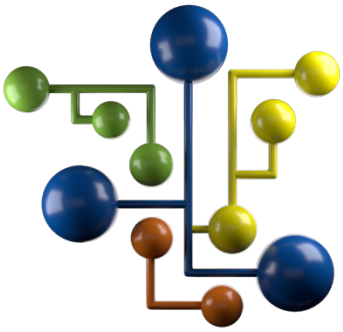

Data Science Academy

www.datascienceacademy.com.br

Introdução à Inteligência Artificial

Extração de Informação

A extração de informação é o processo de aquisição de conhecimento passando os olhos em um texto e procurando por ocorrências de uma classe particular de objetos e de relações entre objetos. A tarefa típica é extrair exemplos de endereços de páginas da Web, com campos de banco de dados de rua, cidade, estado e código postal; ou casos de tempestades a partir de informações meteorológicas, com campos para temperatura, velocidade do vento e precipitação. Em um domínio limitado, isso pode ser feito com alta precisão. Quando o domínio fica mais geral, são necessários modelos linguísticos e técnicas de aprendizagem mais complexas. Veremos mais adiante, como definir modelos de linguagem complexos a partir da estrutura de sintagmas (sintagmas nominais e verbais). Mas até aqui não vimos modelos completos desse tipo; portanto, para as necessidades limitadas de extração de informação, definimos modelos limitados que se aproximam do modelo completo em inglês e se concentram apenas nas partes necessárias para a tarefa à mão.

Descrevemos ainda seis abordagens diferentes para a extração de informações, a fim de aumentar a complexidade em várias dimensões: de determinística a estocástica, de domínio específico a geral, de artesanal a aprendido e de pequena escala a grande escala.

O tipo mais simples de sistema de extração de informação é o sistema de extração baseada em atributos, que assume que todo texto se refere a um único objeto e a tarefa é extrair atributos desse objeto. Por exemplo, considere o problema da extração do texto “IBM ThinkBook 970. Nosso preço: $399,00” como o conjunto de atributos {Fabricante = IBM, Modelo = ThinkBook970 Preço = $ 399,00}. Podemos resolver esse problema através da definição de um modelo (também conhecido como padrão) para cada atributo que gostaríamos de extrair. O modelo é definido por um autômato finito cujo exemplo mais simples é a expressão regular, ou regex (se não faz ideia do que é um regex ou expressão regular, consulte o curso de Python Fundamentos, aqui na DSA). Expressões regulares são usadas em comandos Unix, como grep, em linguagens de programação como Perl e Python e em processadores de texto como o Microsoft Word. Os detalhes variam ligeiramente de uma ferramenta para outra e por isso são mais bem aprendidos através de um manual apropriado, mas aqui vamos mostrar como construir um modelo de expressão regular para preços em dólares:

`[0-9]`	corresponde a qualquer dígito entre 0 e 9
`[0-9]+`	corresponde a um ou mais dígitos
`[.] [0-9] [0-9]`	corresponde a um ponto seguido de dois dígitos
`([.][0-9] [0-9])?`	corresponde a um ponto seguido de dois dígitos ou nada
`[$] [0-9]+([.][0-9] [0-9])?`	corresponde a $249,99 ou $1,23 ou $1.000.000 ou...

Os modelos são geralmente definidos em três partes: um prefixo regex, um objetivo regex e um sufixo regex. Para os preços, o objetivo regex é como anteriormente, o prefixo procura por sequências como “preço:” e o sufixo pode estar vazio. A ideia é que algumas pistas sobre um atributo vem do valor do atributo em si, e alguns vêm a partir do texto circundante. Se uma expressão regular para um atributo corresponder ao texto exatamente uma vez, podemos tirar a parte do texto que é o valor do atributo. Se não houver correspondência, tudo

o que podemos fazer é dar um valor-padrão ou deixar o atributo em falta, mas se houver várias correspondências precisamos de um processo para escolher entre elas. Uma estratégia é ter vários modelos para cada atributo, ordenados por prioridade. Assim, por exemplo, o modelo cuja primeira prioridade é o preço pode procurar pelo prefixo "nosso preço:"; se não for encontrado, procuramos pelo prefixo "preço:" e, se não for encontrado, pelo prefixo vazio. Outra estratégia é pegar todas as correspondências e encontrar alguma maneira de escolher entre elas. Por exemplo, poderíamos pegar o menor preço que está dentro de 50% do preço mais alto. Será selecionado $78,00 como objetivo do texto "Lista de preço $99,00, preços de venda especiais $78,00, transporte $3,00".

Uma etapa acima dos sistemas de extração baseados em atributos estão os sistemas de extração relacional, que lidam com vários objetos e com as relações entre eles. Assim, quando esses sistemas veem o texto "$249,99", eles precisam determinar não apenas que é um preço, mas também que objeto tem esse preço. Um sistema típico baseado em extração relacional é o FASTUS, que lida com histórias de notícias sobre fusões e aquisições corporativas. Pode ser lida a história:

A Bridgestone Sports Co. informou que sexta-feira estabeleceu uma joint venture em Taiwan com a firma local e uma casa de comércio japonesa para produzir tacos de golfe para serem enviados ao Japão. e extrair as relações:

$$e \in JointVentures \wedge Produto\ (e, \text{“clubes de golfe”}) \wedge Data\ (e, \text{“sexta-feira”}) \wedge Membro\ (e, \text{“Bridgestone Sports Co”}) \wedge Membros\ (e, \text{“firma local”}) \wedge Membro\ (e, \text{“uma casa de comércio japonesa”}).$$

Um sistema de extração relacional pode ser construído como uma série de tradutores de estado finito em cascata. Ou seja, o sistema consiste em uma série de pequenos autômatos de estados finitos eficientes (FSAs), em que cada autômato recebe um texto como entrada, traduz o texto em um formato diferente e o passa adiante ao autômato seguinte. O FASTUS consiste em cinco etapas:

1. Tokenização
2. Manuseio de palavras complexas
3. Manuseio de grupos básicos
4. Manuseio de sintagmas complexos
5. Fusão de estruturas

A primeira etapa do FASTUS é a tokenização, que segmenta o fluxo de caracteres em tokens (palavras, números e pontuação). Para o inglês, a tokenização pode ser bastante simples; apenas a separação de caracteres em espaços em branco ou pontuação é um trabalho bastante bom. Alguns tokenizadores também lidam com linguagens de marcação, como HTML, SGML e XML. A segunda fase lida com palavras complexas, incluindo colocações como "estabeleceu" e "joint venture", bem como nomes próprios como "Bridgestone Sports Co.". São

reconhecidos por uma combinação de entradas lexicais e regras de gramática de estado finito. Por exemplo, o nome da empresa pode ser reconhecido pela regra:

PalavraMaiúscula+ ("Company" | "Co" | "Inc" | "Ltd").

A terceira etapa trata de grupos básicos, ou seja, grupos nominais e verbais. A ideia é fatiá-los em unidades que serão geridas pelas fases posteriores. Nesse caso, temos regras simples que apenas se aproximam da complexidade do inglês, porém com a vantagem de ser representáveis por autômatos de estado finito. Um exemplo de sentença iria emergir dessa fase como a seguinte sequência de grupos rotulados:

```
1 NG: Bridgestone Sports Co.      10 NG: uma firma local
2 VG: informou                    11 CJ: e
3 NG: sexta-feira                 12 NG: uma casa de comércio japonesa
4 NG: que                         13 VG: para produzir
5 VG: estabeleceu                 14 NG: tacos de golfe
6 NG: uma joint venture           15 VG: para ser enviado
7 PR: em                          16 PR: ao
8 NG: Taiwan                      17 NG: Japão
9 PR: com
```

NG aqui significa grupo nominal; VG, grupo verbal; PR é preposição e CJ é conjunção. A quarta etapa combina os grupos básicos em frases complexas. Mais uma vez, o objetivo é ter regras que são estados finitos e, portanto, podem ser processadas rapidamente e resultam em frases de saída sem ambiguidade (ou quase sem ambiguidade). Um tipo de regra de combinação trata eventos de domínio específico. Por exemplo, a regra:

Empresa+ Estabeleceu *JointVenture* ("com" Empresa+)?

captura uma maneira de descrever a formação de uma joint venture. Esse estágio é o primeiro na cascata onde a saída é colocada em um modelo de banco de dados, bem como no fluxo de saída. A fase final funde estruturas que foram construídas na etapa anterior. Se a frase seguinte diz "A joint venture vai iniciar a produção em janeiro", nota-se nessa etapa que há duas referências a uma joint venture e que deve ser fundida em uma só. Esse é um exemplo de problema da incerteza de identidade. Em geral, a extração de informações com base em modelo de estado finito funciona bem para um domínio restrito em que é possível predeterminar que assuntos serão discutidos e como eles serão mencionados. O modelo de tradutor em cascata ajuda a modularizar o conhecimento necessário, facilitando a construção do sistema. Esses sistemas funcionam muito bem quando estão em um texto de engenharia reversa que foi gerado por um programa. Por exemplo, um site de compras na Web é gerado por um programa que tira entradas de banco de dados e os formata em páginas da Web; um extrator baseado em modelo, em seguida, recupera o banco de dados original. A extração de

informações de estado finito não é tão bem-sucedida na recuperação de informação em formatos altamente variáveis, tal como um texto escrito por seres humanos sobre uma variedade de assuntos.

Referências:

Livro: Inteligência Artificial
Autor: Peter Norvig