NATURAL LANGUAGE PROCESSING

AULA 2

Prof. Luciano Frontino de Medeiros

CONVERSA INICIAL

Esta aula tem o objetivo de apresentar o conceito de *corpus* e os modelos probabilísticos de linguagem, apresentando também como se estruturam as gramáticas livres de contexto probabilísticas, como se diferenciam das gramáticas estudadas anteriormente, bem como a construção de uma GLCP a partir de um exemplo explicado. Também é apresentada a forma com a qual uma gramática pode ser construída em Python utilizando a NLTK, com os scripts necessários para fazer o processamento de textos a partir de gramáticas probabilísticas.

TEMA 1 – CORPUS E CORPORA

Os métodos e tarefas a serem desempenhadas em processamento de linguagem natural buscam utilizar grandes conjuntos de textos de diversos tipos, sendo essenciais, por exemplo, para a construção de gramáticas baseadas em textos reais, para a aprendizagem de gramáticas probabilísticas e para a validação de processos em recuperação de informação.

Um corpus é uma grande coleção de textos, escritos por seres humanos e para os seres humanos (Russell; Norvig, 2004). Um corpus é utilizado por um software para a execução de tarefas relacionadas com linguagem. Geralmente, a abordagem com corpus está ligada ao uso de estatística e também em tarefas de aprendizagem a partir de textos, como as utilizadas com redes neurais e machine learning. O plural de corpus é corpora.

Exemplos de corpora são a WWW, com os bilhões de textos publicados. Existem corpora construídos com o objetivo de proporcionar grandes bancos de dados de texto. Um destes é o corpus Gutenberg, utilizado na biblioteca NLTK (Natural Language Toolkit) para uso na linguagem Python. Sua origem é o projeto Gutenberg, que disponibiliza para acesso livre em torno de 25.000 livros eletrônicos grátis[1]. O corpus Gutenberg seleciona alguns livros e os disponibiliza para acesso e processamento dentro da biblioteca NLTK (Bird; Klein; Loper, 2019).

A biblioteca NLTK disponibiliza também corpora baseados em textos de chats na web, provenientes de fóruns, discussão sobre livros e filmes. Contém

[1] Disponível em: <http://www.gutenberg.org/>. Acesso em: 1 fev. 2022.

ainda o corpus Brown, criado em 1961, o qual foi o primeiro corpus eletrônico de língua inglesa contendo em torno de 500 textos e que atingia um milhão de palavras[2]. Na Tabela 1, encontra-se um subconjunto de textos que fazem parte do corpus Brown.

Outro corpus presente na biblioteca NLTK é o corpus Reuters. Ele contém 10.788 documentos de notícias totalizando 1,3 milhão de palavras. Os documentos neste corpus são classificados em torno de 90 categorias e agrupados em dois tipos: documentos de treinamento e de teste. Esta divisão tem o objetivo de facilitar a carga dos documentos em algoritmos de aprendizagem de máquina.

Tabela 1 – Alguns textos do corpus Brown

Cód.	Nome	Tipo
A09	**Philadelphia Inquirer**	Reportagem política
	Chicago Tribune	Reportagem política
A10	**Oregonian, Portland**	Reportagem política
A11	**Sun, Baltimore**	Reportagem de esportes
A12	**Dallas Morning News**	Reportagem de esportes
A13	**Rocky Mountain News**	Reportagem de esportes
	Dallas Morning News	Reportagem de esportes
A14	**New York Times**	Reportagem de esportes
A15	**St. Louis Post-Dispatch**	Reportagem de esportes
A16	**Chicago Tribune**	Reportagem da sociedade
A17	**Rocky Mountain News**	Reportagem da sociedade
	Dallas Morning News	Reportagem da sociedade
A18	**Philadelphia Inquirer**	Reportagem da sociedade

Fonte: Elaborado com base em Bird *et al.*, 2019.

Uma característica de muitos corpora é a presença de anotações linguísticas. Estas anotações são colocadas de forma a permitir a orientação a alguma tarefa em especial como, por exemplo, nomes de entidades, estruturas sintáticas e semânticas. Um corpus anotado indica a existência de um trabalho prévio de análise nos textos. Na Tabela 2, encontram-se listados alguns corpora encontrados na biblioteca NLTK.

[2] Disponível em: <http://icame.uib.no/brown/bcm-los.html>. Acesso em: 1 fev. 2022.

Tabela 2 – Alguns textos do corpus Brown

Corpus	Compilador	Conteúdo
Genesis Corpus	Diversas fontes	6 textos, 200k palavras, 6 linguagens
Gutenberg (seleções)	Hart, Newby *et al.*	18 textos, 2M palavras
Inaugural Address Corpus	CSpan	Discursos de posse presidencial dos EUA (1789-presente)
Indian POS-Tagged Corpus	Kumaran *et al.*	60k palavras, anotadas (Bangla, Hindi, Marathi, Telugu)
MacMorpho Corpus	NILC, USP, Brazil	1M palavras anotadas (português brasileiro)
Movie Reviews	Pang, Lee	2k resenhas de filmes com classificação de polaridade de sentimento
Names Corpus	Kantrowitz, Ross	8k nomes masculinos e femininos
NIST 1999 Info Extr (selections)	Garofolo	63k palavras, rede de notícias e marcação SGML de entidade nomeada
Nombank	Meyers	115k proposições, 1400 quadros de substantivos

Fonte: Elaborado com base em Bird *et al.*, 2019.

As anotações em um corpus podem assumir diferentes padrões. Como exemplo, a Figura 1 mostra um excerto do corpus MacMorpho mostrando as anotações (classe gramatical das palavras) colocadas após cada uma das palavras. Na primeira frase, *Um_ART* indica a palavra *Um* como um artigo (“ART” – *article* em inglês) e a palavra *exemplo_N* como um substantivo (“N” – *noun* em inglês).

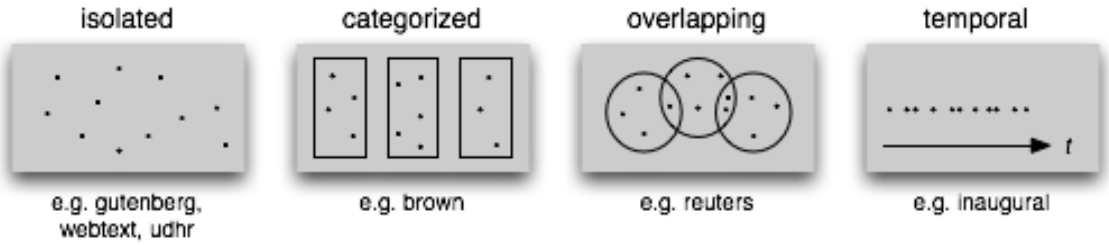
Figura 1 – Excerto do corpus MacMorpho, mostrando quatro frases com anotações após cada palavra, indicando a classe gramatical

```
Um_ART exemplo_N é_V a_ART bacia_N do_PREP+ART Vale_NPROP do_NPROP
Piracicaba_NPROP (_PU a_PREP 170_NUM quilômetros_N a_PREP noroeste_N
de_PREP São_NPROP Paulo_NPROP )_PU ._PU
A_ART terra_N ,_PU que_PRO-KS escoa_V junto_ADV ,_PU chega_V ao_PREP+ART
rio_N e_KC diminui_V o_ART seu_PROADJ leito_N (_PU assoreamento_N )_PU
._PU
Segundo_PREP previsão_N de_PREP Fischer_NPROP ,_PU o_ART quadro_N deve_V
aumentar_V para_PREP 300_NUM criadores_N até_PREP o_ART final_N
deste_PREP+PROADJ ano_N ._PU
Perde_V somente_PDEN para_PREP a_ART Expozebu_NPROP de_NPROP
Uberaba_NPROP (_PU MG_NPROP )_PU e_KC a_ART Expointer_NPROP de_NPROP
Esteio_NPROP (_PU RS_NPROP )_PU ._PU
```

Fonte: Adaptado de Mac-Morpho.

De forma geral, a estrutura mais simples que se pode considerar um corpus é uma coleção de textos sem qualquer estrutura. A partir daí, os textos podem ser agrupados em categorias como gênero, fonte, autoria, linguagem etc. Algumas categorias podem inclusive se sobrepor umas às outras. A Figura 2 esquematiza de forma ilustrada esta caracterização. Desde os textos isolados como o corpus Gutenberg, aos categorizados como o Brown, aos que apresentam categorias em sobreposição como o Reuters, até os que se apresentam temporalmente, como os discursos presidenciais (Bird; Klein; Loper, 2019).

Figura 2 – Estruturas comuns para corpora de textos



Fonte: Elaborado com base em Bird *et al.*, 2019.

A biblioteca NLTK possui métodos com várias funcionalidades para uso com os corpora, indo desde a identificação dos arquivos que contêm os textos, as categorias, o conjunto de palavras e as sentenças presentes. Tais funcionalidades são bastante úteis para verificação do conteúdo e dos textos presentes nos corpora. Conhecer tais métodos é essencial para a execução de tarefas mais avançadas, tendo por base algum corpus específico.

TEMA 2 – MODELOS PROBABILÍSTICOS DE LINGUAGEM

O tema sobre gramáticas apresentado anteriormente mostrou um modelo no qual acontece o processamento lógico das palavras dentro das sentenças. Uma gramática livre de contexto, nessa perspectiva, tem as suas regras elaboradas a partir de uma atividade de planejamento prévio a partir das possíveis formas que uma sentença pode assumir e de que formas ela pode ser analisada. Para modelos pequenos, a elaboração de uma gramática consistente demandará uma engenharia que pode se dar em um tempo razoável. No entanto, para gramáticas que pretendam atender a um largo contexto de sentenças possíveis, tal atividade pode ser muito custosa.

Nesse sentido, os modelos probabilísticos de linguagem assumem que uma palavra ou frase tem uma probabilidade de ocorrência, baseada em uma distribuição calculada a partir de um conjunto de textos. Uma das vantagens desta abordagem é que o modelo probabilístico pode ser treinado a partir dos dados. Em uma gramática elaborada além de um modelo lógico, um algoritmo de aprendizagem leva em conta a ocorrência das palavras que aparecem dentro de um grande conjunto de sentenças. Dessa forma o objetivo de uma gramática construída a partir de um modelo probabilístico permite que as probabilidades associadas às regras possam ser utilizadas com a finalidade de escolha da interpretação mais provável (Russell; Norvig, 2004, p. 807-808).

Um modelo probabilístico pressupõe que as palavras que fazem parte de um léxico possuem uma probabilidade de ocorrência dentro de um corpus. Dado um conjunto de textos C, pode-se expressar uma probabilidade de ocorrência de uma palavra qualquer w dentro deste conjunto por P(w). Como exemplo, supondo uma regra aprendida a partir de um corpus com os seguintes símbolos terminais:

```
Adj -> [big] | [short] | [white] | [blue] | [red]
```

A quantidade total de palavras neste pequeno léxico é de 5. A probabilidade de uma palavra será calculada pela ocorrência dividido pelo total de palavras. Dessa forma, a probabilidade para a palavra *big* será

$$P(\text{"big"}) = \frac{1}{5} = 0.2$$

Modelos que tratam da probabilidade de palavras individuais são denominados de modelos de unigramas. No entanto, modelos baseados em palavras individuais não são tão eficientes em uma análise. Constata-se nas diversas linguagens naturais que certos conjuntos de palavras tendem a aparecer com mais frequência que outros. Modelos que levam em consideração a combinação de duas palavras são chamados de bigramas e de três palavras, trigramas (Russell; Norvig, 2004, p. 808).

Para o caso de bigramas ou trigramas, a definição das probabilidades requer um cálculo ligeiramente diferente, dado que para um bigrama, existe a ocorrência de uma palavra associada a outra palavra. Por exemplo, para calcular a probabilidade de que apareça em um corpus a sequência *big blue*, é

necessário calcular o número de ocorrências em que aparece a palavra *blue*, dado que a palavra *big* aparece antes dela. Portanto, é o caso do cálculo de uma probabilidade condicional. A probabilidade condicional de que um evento B aconteça, dado que aconteça o evento A, é dado pela expressão (Russell; Norvig, 2004, p. 457):

$$P(B|A) = \frac{P(A\ e\ B)}{P(A)}$$

Onde $P(A\ e\ B)$ significa a probabilidade de que aconteça de forma conjunta os eventos A e B.

Por exemplo, se num corpus qualquer contendo 1000 palavras, a palavra *big* apareça 121 vezes e a palavra *blue* apareça em 64 ocorrências, mas que apareçam de forma conjunta em 16 ocorrências. As probabilidades individuais e conjunta são:

$$P(\text{“}big\text{”}) = \frac{121}{1000} = 0{,}121$$

$$P(\text{“}blue\text{”}) = \frac{64}{1000} = 0{,}064$$

$$P(\text{“}big\text{”}\ e\ \text{“}blue\text{”}) = \frac{16}{1000} = 0{,}016$$

A probabilidade condicional da palavra *blue* aparecer, dado que aparece a palavra *big* é, portanto,

$$P(\text{“}blue\text{”} \mid \text{”}big\text{”}) = \frac{P(\text{”}big\text{”}\ e\ \text{”}blue\text{”})}{P(\text{”}big\text{”})} = \frac{0{,}016}{0{,}121} \cong 0{,}132$$

Pode-se constatar também que o cálculo da probabilidade condicional pode ser simplificado, dividindo-se a quantidade de ocorrências da forma conjunta (16) pela quantidade de ocorrências da palavra *big* (121). O cálculo da probabilidade para bigramas pode ser estendido para trigramas e assim sucessivamente. Por exemplo, para o trigrama *the big blue*, caso a ocorrência seja de uma vez no conjunto considerado anteriormente, tem-se:

$$P(\text{”}the\text{”}\ e\ \text{“}big\text{”}\ e\ \text{“}blue\text{”}) = \frac{1}{1000} = 0{,}001$$

A probabilidade condicional para a ocorrência da palavra the, dado que ocorrem as palavras *big* e *blue*, é:

$$P(\text{“}the\text{”} \mid \text{”}big\text{”}\ e\ \text{"}blue\text{"}) = \frac{P(\text{"the" e "big" e "blue"})}{P(\text{"big" e "blue"})} = \frac{0{,}001}{0{,}016} \cong 0{,}063$$

É provável que a maior parte dos bigramas, por combinarem duas palavras que são altamente improváveis de acontecer, retornem uma contagem de ocorrências igual a zero. Isto pode ser mais provável ainda de ocorrer com trigramas, zerando, portanto, as probabilidades. A solução para este problema é utilizar o recurso da suavização de soma um (Russell; Norvig, 2004, p. 808). A ideia é somar um à contagem de cada um dos bigramas possíveis. Para N palavras, B bigramas possíveis e o número de ocorrências c, o cálculo da probabilidade pode ser modificado para:

$$P(w) = \frac{c+1}{N+B}$$

Dessa maneira, ainda que o número de ocorrências seja zero, a probabilidade terá um valor bem pequeno, porém diferente de zero. Por exemplo, para um conjunto de 100 palavras gera um total de $100 * 99 / 2 = 4950$ bigramas[3]. Com $N = 100$ e $B = 4950$, caso o número de ocorrências de um bigrama específico seja zero, a probabilidade neste caso fica:

$$P(w) = \frac{o+1}{N+B} = \frac{0+1}{100+4950} = \frac{1}{5050} \cong 0{,}000198$$

Outra forma de solucionar o problema da probabilidade zero é a suavização por interpolação linear. Aqui, utiliza-se os valores das probabilidades para a ocorrência de unigramas, bigramas e trigramas para elaborar uma estimativa de probabilidade. Para os parâmetros c_1, c_2 e c_3, tem-se a seguinte fórmula de interpolação:

$$P(w_i \mid w_{i-2}\, w_{i-1}) = c_1\, P(w_i) + c_2\, P(w_i \mid w_{i-1}) + c_3\, P(w_i \mid w_{i-2}\, w_{i-1})$$

[3] Da análise combinatória, utiliza-se a fórmula da combinação: $C_2^{100} = \frac{n!}{p!(n-p)!} = \frac{100!}{2!98!} = \frac{100.99}{2} = 4950$.

Onde considera-se, por questões de normalização, $c_1 + c_2 + c_3 = 1$. Os parâmetros c_i podem ter valores fixados ou ainda serem determinados por algoritmo de aprendizagem.

TEMA 3 – GRAMÁTICAS PROBABILÍSTICAS

Anteriormente, foram estudadas as gramáticas livres de contexto para a análise de sentenças, conforme uma determinada linguagem. Uma gramática contém um conjunto de regras de sintaxe que permitem a análise da frase, decompondo sucessivamente até alcançar os símbolos terminais. Em uma gramática livre de contexto probabilística (também chamada de *GLCP*), cada regra contém uma probabilidade associada. Desta forma, a soma das probabilidades das regras que contém o mesmo lado esquerdo deve ser 1 (Russell; Norvig, 2004, p. 810). Às gramáticas que não contém probabilidades, pode-se dizer que constituem gramáticas livres de contexto determinísticas (GLCD).

O problema com a análise de bigramas ou trigramas visto anteriormente (ou, por extensão, n-gramas para n grupos de palavras) é que essa se limita localmente aos grupos compostos de palavras que se combinam em sequência, não considerando possíveis relações gramaticais que podem ocorrer entre palavras além da distância n dos n-gramas.

Uma gramática probabilística pode ser construída por meio de aprendizagem. As palavras que ocorrem nas sentenças de um texto ou corpus são contadas, a partir de uma análise sintática executada com uma gramática determinística (que pode, inclusive, ser originada a partir de outro corpus). Considerando uma gramática determinística para ilustrar a obtenção de uma gramática probabilística, o exemplo anterior é adaptado de forma a constituir uma gramática determinística mostrada na Figura 3.

Figura 3 – Uma gramática simples para língua inglesa

Regras	
1	Sentença → SN SV
2	SN → Substantivo | Adjetivo | Artigo Substantivo | Artigo Adjetivo Substantivo
3	SV → Verbo | Verbo SN | Verbo SN Advérbio | Verbo Preposição SN
4	Substantivo → 'house' | 'cat' | 'water' | 'dog' | 'John' | 'Mary'
5	Artigo → 'A' | 'a' | 'The' | 'the'
6	Adjetivo → 'big' | 'small' | 'white' | 'black' | 'tall' | 'short'
7	Verbo → 'goes' | 'comes' | 'fights' | 'drinks' | 'loves' | 'has' | 'is' | 'are' | 'hates'
8	Preposição → 'with' | 'before' | 'after' | 'until' | 'of' | 'since'

Fonte: Elaborado com base em Forouzan e Mosharraf, 2011.

Portanto, a construção de uma GLCP envolve a combinação da construção de uma GLCD mais o cálculo de probabilidades associado com o corpus utilizado para a aprendizagem. Uma GLCP resulta em mais robustez, à medida que contenha milhares ou milhões de palavras ou sentenças. Um corpus anotado é essencial para este tipo de tarefa.

Na Figura 4, constam 4 sentenças que foram analisadas de acordo com esta gramática. A análise mostra a árvore sintática da sentença com os símbolos expressos entre parênteses. Cada uma das sentenças produziu uma árvore de acordo com a gramática da Figura 3.

Figura 4 – Análise das sentenças

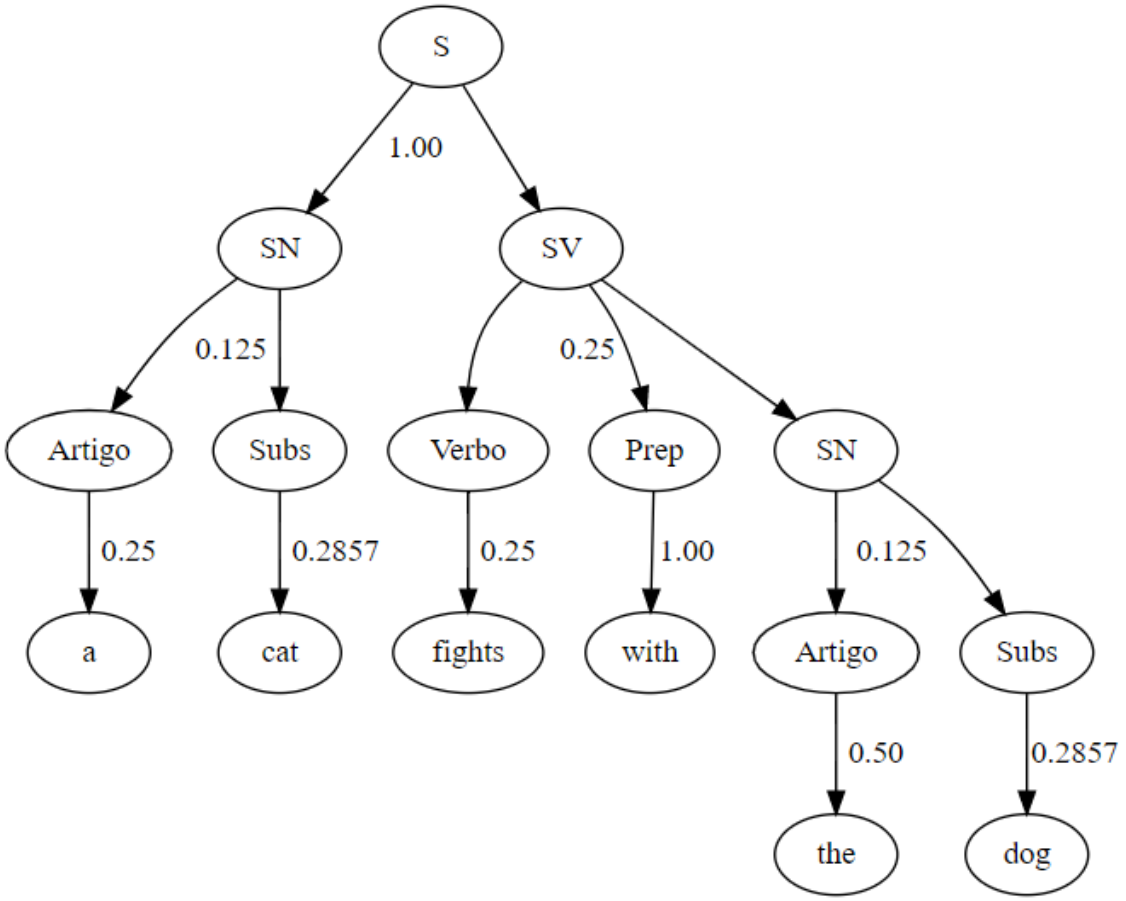
No.	Sentença	Análise
1	'a small cat loves the white dog'	(S (SN (Artigo a) (Adjetivo small) (Substantivo cat)) (SV (Verbo loves) (SN (Artigo the) (Adjetivo white) (Substantivo dog))))
2	'John loves Mary'	(S (SN (Substantivo John)) (SV (Verbo loves) (SN (Substantivo Mary))))
3	'The cat fights with the black dog'	(S (SN (Artigo The) (Substantivo cat)) (SV (Verbo fights) (Preposição with) (SN (Artigo the) (Adjetivo black) (Substantivo dog))))
4	'John is tall'	(S (SN (Substantivo John)) (SV (Verbo is) (SN (Adjetivo tall))))

A gramática probabilística obtida a partir do aprendizado com estas 4 sentenças é mostrada na Figura 5. Pode-se notar que a gramática contém símbolos terminais e não terminais apenas para os casos que ocorreram, a partir das sentenças utilizadas para treinamento, conforme a Figura 4. Em cada regra, a soma total das probabilidades deve, portanto, ser 1. Por exemplo, o verbo (conjugação verbal) *loves* aparece em duas sentenças ($2/4 = 0.50$), enquanto que o verbo *fights* aparece em uma ($1/4 = 0.25$) e, por fim, o verbo 'is' aparece também em uma ($1/4 = 0.25$); No total, $2/4 + 1/4 + 1/4 = 1$.

Figura 5 – Gramática probabilística gerada a partir das sentenças da Figura 4 e da gramática determinística da Figura 3

	Regras
1	Sentença → SN SV [1.0]
2	SN → Substantivo [0.375] | Adjetivo [0.125] | Artigo Substantivo [0.125] | Artigo Adjetivo Substantivo [0.375]
3	SV → Verbo SN [0.75] | Verbo Preposição SN [0.25]
4	Substantivo → 'cat' [0.28] | [water] | 'dog' [0.28] | 'John' [0.28] | 'Mary' [0.14]
5	Artigo → 'a' [0.25] | 'The' [0.25] | 'the' [0.50]
6	Adjetivo → 'small' [0.25] | 'white' [0.25] | 'black' [0.25] | 'tall' [0.25]
7	Verbo → 'fights' [0.25] | 'loves' [0.50] | 'is' [0.25]
8	Preposição → 'with' [1.0]

Figura 6 – Árvore sintática da frase a *cat fights with the dog* com as probabilidades associadas, conforme a GLCP da Figura 5



Como exemplo, para a análise sintática da sentença *a cat fights with the dog*, cuja árvore é mostrada na Figura 6, o processo de geração inclui cada probabilidade associada, buscando maximizar a informação. Um algoritmo utilizado para este cálculo é o algoritmo de Viterbi. Este algoritmo busca armazenar o cálculo de probabilidades feito para as ramificações atuais, de maneira a ser utilizado para os cálculos seguintes para a probabilidade da sentença (Russell; Norvig, 2004, p. 810).

A probabilidade resultante para esta sentença é calculada, portanto, a partir da multiplicação das probabilidades de cada ramificação. Portanto, a probabilidade será

$$P(\text{“}a\ cat\ fights\ with\ the\ dog\text{”}) = 1.00 \times 0.125 \times 0.25 \times 0.2857 \times 0.25 \times 0.25 \times 1.00 \times 0.125 \times 0.50 \times 0.2857 = 9.9649.10^{-6}$$

Um dos problemas das GLCP é de que são livres de contexto. Por isso, elas não consideram outros termos que podem causar conflitos semânticos na

análise de uma sentença. Neste caso, podem ser utilizadas GLCP lexicalizadas, onde a palavra-chave dentro de uma sentença desempenhe um papel importante para o cálculo da probabilidade. Uma GLCP lexicalizada, na verdade, já não é mais tão “livre de contexto”, ficando mais aproximada ao conjunto de palavras (léxico) que disponibiliza então informação adicional para o cálculo das probabilidades. Outro problema é que as GLCPs tendem a ter uma preferência por sentenças muito curtas, pois com poucas palavras, o cálculo da probabilidade tende a resultar em valores maiores (Russell; Norvig, 2004, p. 812). Por exemplo, a sentença *a cat* produziria uma probabilidade igual a 0.125, muito mais alta que a calculada no exemplo da Figura 6.

TEMA 4 – USO DA NLTK COM GRAMÁTICAS LIVRES DE CONTEXTO PROBABILÍSTICAS

O Natural Language Toolkit (NLTK) foi originalmente criado em 2001 como parte de um curso de linguística computacional no Departamento de Ciência da Computação e da Informação da Universidade da Pensilvânia. Desde então, ele foi desenvolvido e expandido com a ajuda de dezenas de colaboradores. Já foi adotado em cursos em dezenas de universidades e serve como base para muitos projetos de pesquisa (Bird; Klein; Loper, 2019).

O NLTK foi projetado com quatro objetivos principais em mente.

- Simplicidade: para fornecer uma estrutura intuitiva junto com blocos de construção substanciais, dando aos usuários um conhecimento prático de PLN sem ficar atolado na tediosa manutenção geralmente associada ao processamento de dados de linguagem anotada.
- Consistência: Para fornecer uma estrutura uniforme com interfaces e estruturas de dados consistentes e nomes de métodos facilmente adivinhados.
- Extensibilidade: Para fornecer uma estrutura na qual novos módulos de software possam ser facilmente acomodados, incluindo implementações alternativas e abordagens concorrentes para a mesma tarefa.
- Modularidade: Para fornecer componentes que podem ser usados de forma independente, sem a necessidade de entender o resto do toolkit.

Na Tabela 3, estão algumas tarefas de processamento de linguagem e módulos NLTK correspondentes com exemplos de funcionalidade (Bird; Klein; Loper, 2019).

Tabela 3 – Exemplos de funcionalidade

Tarefa	Módulo	Funcionalidade
Acesso a corpora	`corpus`	interfaces padronizadas para corpora e léxicos
Processamento de palavras	`tokenize, stem`	tokenizers, tokenizers de sentenças, stemmers
Descoberta de collocations	`collocations`	informações mútuas de teste t, qui-quadrado e ponto-a-ponto
Marcação de parte da fala (part-of-speech)	`tag`	n-grama, HMM
Machine learning	`classify, cluster, tbl`	árvore de decisão, entropia máxima, naive Bayes, k-médias
Chunking	`chunk`	expressão regular, n-grama, entidade nomeada
Parsing	`parse, ccg`	gráfico, baseado em recursos, unificação, probabilística, dependência

No Jupyter Notebook, a NLTK pode ser acessada com os seguintes comandos:

```
import nltk
```

Para a utilização de gramáticas, deve-se importar o pacote CFG:

```
from nltk import CFG
```

Uma gramática pode ser carregada a partir de uma string contendo as regras no formato BNF, utilizando o comando CFG.fromstring. Por exemplo, a gramática da figura 3 pode ser carregada com o seguinte comando:

```
grammar = CFG.fromstring("""
  S -> SN SV
  SN -> Substantivo | Adjetivo | Artigo Substantivo | Artigo Adjetivo Substantivo
  SV -> Verbo | Verbo SN | Verbo SN Advérbio | Verbo Preposição SN
  Substantivo -> 'house' | 'cat' | 'water' | 'dog' | 'John' | 'Mary'
  Artigo -> 'A' | 'a' | 'The' | 'the'
  Adjetivo -> 'big' | 'small' | 'white' | 'black' | 'tall' | 'short'
  Verbo -> 'goes' | 'comes' | 'fights' | 'drinks' | 'loves' | 'has' | 'is' | 'are' | 'hates'
  Preposição -> 'with' | 'before' | 'after' | 'until' | 'of' | 'since'
```

```
""")
```

Com a gramática colocada na variável grammar, ela pode ser visualizada a partir do comando print:

```
print('Gramática:', grammar)
```

Este comando apresenta a seguinte saída, indicando 40 produções, com o símbolo inicial sendo “S”:

```
Gramática: Grammar with 40 productions (start state = S)
    S -> SN SV
    SN -> Substantivo
    SN -> Adjetivo
    SN -> Artigo Substantivo
    SN -> Artigo Adjetivo Substantivo
    SV -> Verbo
    SV -> Verbo SN
    SV -> Verbo SN Advérbio
    SV -> Verbo Preposição SN
    Substantivo -> 'house'
    Substantivo -> 'cat'
    Substantivo -> 'water'
    Substantivo -> 'dog'
    Substantivo -> 'John'
    Substantivo -> 'Mary'
    Artigo -> 'A'
    Artigo -> 'a'
    Artigo -> 'The'
    Artigo -> 'the'
    Adjetivo -> 'big'
    Adjetivo -> 'small'
    Adjetivo -> 'white'
    Adjetivo -> 'black'
    Adjetivo -> 'tall'
    Adjetivo -> 'short'
    Verbo -> 'goes'
    Verbo -> 'comes'
    Verbo -> 'fights'
    Verbo -> 'drinks'
    Verbo -> 'loves'
    Verbo -> 'has'
    Verbo -> 'is'
    Verbo -> 'are'
    Verbo -> 'hates'
    Preposição -> 'with'
    Preposição -> 'before'
    Preposição -> 'after'
    Preposição -> 'until'
    Preposição -> 'of'
    Preposição -> 'since'
```

Pode-se exibir especificamente qual o símbolo inicial da gramática com **start()**

```
print('grammar.start()   =>', grammar.start())
```

Mostrando a seguinte saída para o símbolo inicial:

```
grammar.start()    => S
```

As produções podem ser colocadas em uma lista do Python com **productions()**

```
print(grammar.productions())
```

Com a saída característica no formato de lista:

```
[S -> SN SV, SN -> Substantivo, SN -> Adjetivo, SN -> Artigo Substa
ntivo, SN -> Artigo Adjetivo Substantivo, SV -> Verbo, SV -> Verbo
SN, SV -> Verbo SN Advérbio, SV -> Verbo Preposição SN, Substantivo
-> 'house', Substantivo -> 'cat', Substantivo -> 'water', Substanti
vo -> 'dog', Substantivo -> 'John', Substantivo -> 'Mary', Artigo -
> 'A', Artigo -> 'a', Artigo -> 'The', Artigo -> 'the', Adjetivo ->
'big', Adjetivo -> 'small', Adjetivo -> 'white', Adjetivo -> 'black
', Adjetivo -> 'tall', Adjetivo -> 'short', Verbo -> 'goes', Verbo
-> 'comes', Verbo -> 'fights', Verbo -> 'drinks', Verbo -> 'loves',
Verbo -> 'has', Verbo -> 'is', Verbo -> 'are', Verbo -> 'hates', Pr
eposição -> 'with', Preposição -> 'before', Preposição -> 'after',
Preposição -> 'until', Preposição -> 'of', Preposição -> 'since']
```

Para verificar se um conjunto de símbolos faz parte de uma gramática, pode-se utilizar **check_coverage()** (deve-se utilizar em estrutura try-except, pois caso não exista a palavra, é gerada uma exceção).

```
print(grammar.check_coverage(['a','cat']))
```

Uma das estruturas bastante úteis da NLTK é o módulo Tree, que permite a geração de árvores sintáticas a partir de gramáticas definidas. O script a seguir permite a criação de uma lista de termos a partir de uma frase, cria um objeto ChartParser e depois constrói a árvore a partir da lista em um loop no objeto ChartParser.

```
from nltk import Tree
sent = 'a small cat loves the white dog'.split()
parser = nltk.ChartParser(grammar)
for tree in parser.parse(sent):
   print(tree)
```

O script produz a seguinte saída, com a árvore sintática gerada, utilizando a notação de parênteses:

```
(S
  (SN (Artigo a) (Adjetivo small) (Substantivo cat))
  (SV
    (Verbo loves)
```

```
    (SN (Artigo the) (Adjetivo white) (Substantivo dog))))
```

TEMA 5 – APRENDIZAGEM PARA A GRAMÁTICA

Para a aprendizagem da gramática probabilística, as sentenças deverão estar neste formato. Portanto, será criada uma lista com as sentenças e um script que faça o processamento das listas em árvores sintáticas. As sentenças são divididas (*splitted*) em palavras e depois são construídas as árvores de análise. Em seguida, é criado um objeto ChartParser para cada sentença e depois armazenado na lista de treinamento (*train*).

```
data = ['a small cat loves the white dog',
        'John loves Mary',
        'The cat fights with the black dog',
        'John is tall']
train = []
for s in data:
   print("Sentença:",s)
   sent = s.split()
   print(sent)
   parser = nltk.ChartParser(grammar)
   for tree in parser.parse(sent):
      temp = tree
   print("Análise: ",temp)
   train.append(Tree.fromstring(str(temp)))
print("Árvores sintáticas:")
print(train)
```

O resultado mostra o conteúdo do processamento:

```
Sentença: a small cat loves the white dog
['a', 'small', 'cat', 'loves', 'the', 'white', 'dog']
Análise:  (S
  (SN (Artigo a) (Adjetivo small) (Substantivo cat))
  (SV
    (Verbo loves)
    (SN (Artigo the) (Adjetivo white) (Substantivo dog))))

Sentença: John loves Mary
['John', 'loves', 'Mary']
Análise:  (S
  (SN (Substantivo John))
  (SV (Verbo loves) (SN (Substantivo Mary))))

Sentença: The cat fights with the black dog
['The', 'cat', 'fights', 'with', 'the', 'black', 'dog']
Análise:  (S
  (SN (Artigo The) (Substantivo cat))
  (SV
    (Verbo fights)
    (Preposição with)
    (SN (Artigo the) (Adjetivo black) (Substantivo dog))))
```

```
Sentença: John is tall
['John', 'is', 'tall']
Análise:  (S (SN (Substantivo John)) (SV (Verbo is) (SN (Adjetivo t
all))))


Árvores sintáticas:
[Tree('S', [Tree('SN', [Tree('Artigo', ['a']), Tree('Adjetivo', ['s
mall']), Tree('Substantivo', ['cat'])]), Tree('SV', [Tree('Verbo',
['loves']), Tree('SN', [Tree('Artigo', ['the']), Tree('Adjetivo', [
'white']), Tree('Substantivo', ['dog'])])])]), Tree('S', [Tree('SN'
, [Tree('Substantivo', ['John'])]), Tree('SV', [Tree('Verbo', ['lov
es']), Tree('SN', [Tree('Substantivo', ['Mary'])])])]), Tree('S', [
Tree('SN', [Tree('Artigo', ['The']), Tree('Substantivo', ['cat'])])
, Tree('SV', [Tree('Verbo', ['fights']), Tree('Preposição', ['with'
]), Tree('SN', [Tree('Artigo', ['the']), Tree('Adjetivo', ['black']
), Tree('Substantivo', ['dog'])])])]), Tree('S', [Tree('SN', [Tree(
'Substantivo', ['John'])]), Tree('SV', [Tree('Verbo', ['is']), Tree
('SN', [Tree('Adjetivo', ['tall'])])])])]
```

Com a lista de treinamento pronta, faz-se agora a verificação das produções existentes. Cada produção é como se fosse uma regra simplificada, que auxiliará no cálculo das probabilidades para a GLCP. Esta simplificação é feita com os métodos **collapse_unary()**. As regras também ficarão na forma normal de Chomsky, a partir do método **chomsky_normal_form()**. Cada produção é adicionada em uma nova lista.

```
Productions = []
for tree in train:
   # Modifica os ramos A-B-C em A-B+C:
   tree.collapse_unary(collapsePOS = False)
   # Modifica A->(B,C,D) em A->B,C+D->D
   tree.chomsky_normal_form(horzMarkov = 2)
   productions += tree.productions()
productions
```

A lista de produções então, contém:

```
[S -> SN SV,
 SN -> Artigo SN|<Adjetivo-Substantivo>,
 Artigo -> 'a',
 SN|<Adjetivo-Substantivo> -> Adjetivo Substantivo,
 Adjetivo -> 'small',
 Substantivo -> 'cat',
 SV -> Verbo SN,
 Verbo -> 'loves',
 SN -> Artigo SN|<Adjetivo-Substantivo>,
 Artigo -> 'the',
 SN|<Adjetivo-Substantivo> -> Adjetivo Substantivo,
 Adjetivo -> 'white',
 Substantivo -> 'dog',
 S -> SN SV,
 SN -> Substantivo,
 Substantivo -> 'John',
 SV -> Verbo SN,
 Verbo -> 'loves',
```

```
SN -> Substantivo,
Substantivo -> 'Mary',
S -> SN SV,
SN -> Artigo Substantivo,
Artigo -> 'The',
Substantivo -> 'cat',
SV -> Verbo SV|<Preposição-SN>,
Verbo -> 'fights',
SV|<Preposição-SN> -> Preposição SN,
Preposição -> 'with',
SN -> Artigo SN|<Adjetivo-Substantivo>,
Artigo -> 'the',
SN|<Adjetivo-Substantivo> -> Adjetivo Substantivo,
Adjetivo -> 'black',
Substantivo -> 'dog',
S -> SN SV,
SN -> Substantivo,
Substantivo -> 'John',
SV -> Verbo SN,
Verbo -> 'is',
SN -> Adjetivo,
Adjetivo -> 'tall']
```

A partir de agora, pode-se obter a gramática probabilística por indução. Serão utilizados os módulos induce_pcfg. Este módulo contém **induce_pcfg()**, que calcula as probabilidades. Também é utilizado o módulo **Nonterminal** para indicar o símbolo inicial como um símbolo não terminal, com uma função de mesmo nome.

```
from nltk import induce_pcfg
from nltk import Nonterminal
S = Nonterminal('S')
prob_grammar = induce_pcfg(S, productions)
print(prob_grammar)
```

Por fim, tem-se a gramática probabilística, com os valores das probabilidades ao lado de cada regra:

```
Grammar with 24 productions (start state = S)
    S -> SN SV [1.0]
    SN -> Artigo SN|<Adjetivo-Substantivo> [0.375]
    Artigo -> 'a' [0.25]
    SN|<Adjetivo-Substantivo> -> Adjetivo Substantivo [1.0]
    Adjetivo -> 'small' [0.25]
    Substantivo -> 'cat' [0.285714]
    SV -> Verbo SN [0.75]
    Verbo -> 'loves' [0.5]
    Artigo -> 'the' [0.5]
    Adjetivo -> 'white' [0.25]
    Substantivo -> 'dog' [0.285714]
    SN -> Substantivo [0.375]
    Substantivo -> 'John' [0.285714]
    Substantivo -> 'Mary' [0.142857]
    SN -> Artigo Substantivo [0.125]
    Artigo -> 'The' [0.25]
    SV -> Verbo SV|<Preposição-SN> [0.25]
    Verbo -> 'fights' [0.25]
```

```
SV|<Preposição-SN> -> Preposição SN [1.0]
Preposição -> 'with' [1.0]
Adjetivo -> 'black' [0.25]
Verbo -> 'is' [0.25]
SN -> Adjetivo [0.125]
Adjetivo -> 'tall' [0.25]
```

Com a GLCP, é possível agora calcular as probabilidades de sentenças diferentes ocorrerem a partir desta gramática. Como exemplo, para a sentença *a cat fights with the dog*, deve ser calculada a probabilidade utilizando-se o algoritmo de Viterbi mencionado anteriormente. Utiliza-se o objeto ViterbiParser, gera-se a árvore sintática, porém calculando agora a probabilidade resultante (compare com o valor obtido em tema anterior):

```
sample = 'a cat fights with the dog'
test = sample.split()
viterbi_parser = nltk.ViterbiParser(prob_grammar)
for tree in viterbi_parser.parse(test):
   print(tree)
```

O resultado será:

```
(S
  (SN (Artigo a) (Substantivo cat))
  (SV
    (Verbo fights)
    (SV|<Preposição-SN>
      (Preposição with)
      (SN (Artigo the) (Substantivo dog))))) (p=9.96492e-06)
```

FINALIZANDO

Nesta aula, foram estudados o conceito de *corpus* e *corpora*, e também os modelos probabilísticos de linguagem. Dentro dos modelos, foi abordado o uso de gramáticas livres de contexto probabilísticas, uma alternativa para fazer o processamento de sentenças apresentando, além da análise sintática propriamente, uma probabilidade que auxilia na redução de conflitos de interpretação sintática, resultando naquela mais provável. A elaboração de uma GLCP foi demonstrada utilizando também a NLTK, uma toolkit para o Python própria para processamento de linguagem natural, demonstrando-se um conjunto de scripts a construção de uma GLCP, sua aprendizagem e também o cálculo de probabilidades para o teste de novas sentenças.

REFERÊNCIAS

BIRD, S.; KLEIN, E.; LOPER, E. **Natural Language Processing with Python**. [S. l.], 2019. Disponível em: <http://www.nltk.org/book/>. Acesso em: 10 ago. 2021.

FOROUZAN, B.; MOSHARRAF, F. **Fundamentos da Ciência da Computação**. São Paulo: Cengage Learning, 2011.

RUSSELL, S.; NORVIG, P. **Inteligência Artificial**. Rio de Janeiro: Campus, 2004.