Processamento de Linguagem Natural

Aula 5: Part-of-Speech (POS) Tagging com modelos n-grama

*Objetivo: ao fim desta aula, o aluno será capaz de usar n-gramas e modelos pré-treinados na tarefa de rotular palavras de acordo com suas classes morfológicas e aplicar esta técnica na melhoria de análises textuais.*

|  |
| --- |
| **Exercício 1** |
| *Objetivo: usar análise morfológica para construir frases* |
| Encontre todos os substantivos, adjetivos, verbos e advérbios no trecho abaixo:  *Hércules derrotou brilhantemente o terrível Leão de Neméia.*  (a) Que outras classes morfológicas de palavras podemos encontrar aqui?  (b) Que palavras poderiam ser removidas do texto de forma que a história ainda fosse contada? Que classes morfológicas foram removidas?  (c) Como poderíamos mudar a frase de forma que fosse atribuída a outra pessoa o crédito pela derrota do leão? Que classes morfológicas foram afetadas?  (d) Como poderíamos mudar a frase de forma que o desfecho da história não fosse de conflito, e sim de paz e harmonia? Que classes morfológicas foram afetadas?  (e) Seria possível manter a sequência de classes morfológicas desta frase e escrever uma história completamente diferente? Dê um exemplo. |

|  |
| --- |
| **Exercício 2** |
| *Objetivo: usar um tagger no nltk* |
| A maneira mais simples de atribuir rótulos de classes morfológicas a palavras é simplesmente assumir que todas as palavras são substantivos. Veja, por exemplo, neste código:  from nltk.tag import DefaultTagger  default\_tagger = DefaultTagger('N')  frase = "uma frase qualquer".split()  default\_tagger.tag(frase)  (a) O que significa a constante ‘N’ no default tagger?  (b) O que faz o método “tag”?  (c) Qual é o formato da entrada do método “tag”?  (d) Qual é o formato da saída do método “tag”? |

|  |
| --- |
| **Exercício 3** |
| *Objetivo: abrir uma base de dados de pos\_tagging e avaliar o tagger* |
| É evidentemente pouco prático testar nosso rotulador manualmente, de frase em frase. Neste exercício, vamos realizar a avaliação de nosso rotulador.  (a) Abra o notebook “pos\_tagging\_macmorpho”.  (b) Abra a base de dados “macmorpho\_test”.  (c) Como ela está organizada?  (d) Divida a base de dados em frases (dica: seria melhor usar re.split() ou re.findall())?  (e) Divida as frases em *tokens*, como se tivessem sido geradas pelo método .tag() do rotulador  (f) Use o método .evaluate() para avaliar o índice de acertos de nosso rotulador “default”.  (g) Qual é o índice de acertos se nosso rótulo padrão for “verbo” (‘V’)? Faça o teste também com outros rótulos. |

|  |
| --- |
| **Exercício 4** |
| *Objetivo: relacionar palavras a suas classes morfológicas, fora de contexto, e discutir as ambiguidades que podem surgir* |
| (a) Encontre as classes morfológicas das palavras abaixo:  Herói | Veloz | Cantarei | Lentamente | Cantar | Planta | Vela | Fazer | Fantasia | Verde  (b) As decisões que você tomou são não-ambíguas?  (c) Mesmo havendo ambiguidade, existe alguma decisão que, a prior, seria mais “segura”?  (d) Como poderíamos tomar decisões para palavras isoladas de forma a termos uma probabilidade mais alta de acertarmos? |

|  |
| --- |
| **Exercício 5** |
| *Objetivo: usar o modelo “unigrama” para encontrar classes morfológicas* |
| Na biblioteca NLTK, podemos usar um tagger baseado em maximizar a probabilidade de acerto em unigramas. Ele é treinado em uma base de dados de treino, que deve ser diferente da base de dados de teste:  from nltk.tag import UnigramTagger  unigram\_tagger = UnigramTagger(s)  unigram\_tagger.evaluate(s\_)  (a) Por que a base de dados de treino deve ser diferente da base de teste?  (b) Abra e pré-processe, para treino, a base de dados macmorpho-train. Qual foi o índice de acertos que obtivemos na base de teste?  (c) Há palavras que aparecem na base de teste, mas não aparecem na base de treino. Qual seria o problema disso na avaliação de probabilidades do UnigramTagger? Como poderíamos resolver essa situação?  (d) O que significa “backoff” em:  unigram\_tagger = UnigramTagger(s, backoff=default\_tagger)  (e) Como o índice de acertos é afetado quando usamos o DefaultTagger como backoff? |

|  |
| --- |
| **Exercício 2** |
| *Objetivo: abrir uma base de dados para identificação de classes morfológicas* |
| Abra o notebook “pos\_tagging\_macmorpho”.  Abra a base de dados “macmorpho“.  (a) Como ela está organizada?  (b) Divida a base de dados em frases (dica: seria melhor usar re.split() ou re.findall())?  (c) Divida as frases em *tokens*, sendo cada *token* uma tupla (palavra, rótulo de classe\_morfologica). Utilize expressões regulares ou veja os elementos do nltk que podem ajudar. Ao fim deste processo, deveríamos ter alguma coisa como uma lista de frases, sendo que cada frase é uma lista de tuplas, e cada uma dessas tuplas é um par palavra + rótulo:  [[('jersei', 'N'), ('atinge', 'V'), ('média', 'N'), ('de', 'PREP'), ('cr$', 'CUR'), ('1,4', 'NUM'), ('milhão', 'N'), ('na', 'PREP+ART'), ('venda', 'N'), ('da', 'PREP+ART'), ('pinhal', 'NPROP'), ('em', 'PREP'), ('são', 'NPROP'), ('paulo', 'NPROP')], [('programe', 'V'), ('sua', 'PROADJ'), ('viagem', 'N'), ('à', 'PREP+ART'), ('exposição', 'NPROP'), ('nacional', 'NPROP'), ('do', 'NPROP'), ('zebu', 'NPROP'), (',', 'PU'), ('que', 'PRO-KS'), ('começa', 'V'), ('dia', 'N'), ('25', 'N')]] |

|  |
| --- |
| **Exercício 2** |
| *Objetivo: usar o conceito de classes morfológicas para gerar novos textos* |
| Considere o trecho:  “Hércules é um herói conhecido por ser forte”  Quais das palavras abaixo poderiam substituir a palavra “forte” e ainda manter a frase gramaticalmente correta?  fraco | gol | heroicamente | ornitorrinco | pensou | guarda-chuvas | pipoca | semi-deus | grego | alto | azul | sorte | azar | monitor | telefone | transparente | disse | olhava | olhos  Qual foi o critério que você usou para isso? |

|  |
| --- |
| **Exercício 3** |
| *Objetivo: relacionar palavras a suas classes morfológicas, fora de contexto* |
| Encontre as classes morfológicas das palavras abaixo:  (a) Herói  (b) Veloz  (c) Cantarei  (d) Lentamente  (e) Cantar  (f) Planta  (g) Vela  (h) Fazer |

|  |
| --- |
| **Exercício 4** |
| *Objetivo: abrir uma base de dados para identificação de classes morfológicas* |
| Abra o notebook “pos\_tagging\_macmorpho”.  Abra a base de dados “macmorpho“.  (a) Como ela está organizada?  (b) Divida a base de dados em frases  (c) Divida as frases em *tokens*, sendo cada *token* uma tupla (palavra, classe\_morfologica). Utilize expressões regulares ou veja os elementos do nltk que podem ajudar.  (d) Prossiga para o exercício seguinte. |

|  |
| --- |
| **Exercício 5** |
| *Objetivo: avaliar o uso de unigramas para classificação de classes morfológicas em frases* |
| No nltk, temos acesso a um tagger que usa unigramas. No código abaixo, assumimos que s contém uma base de dados para treino do modelo, organizada como está abaixo.  # s[frase][palavra] = (palavra, tipo)  from nltk.tag import UnigramTagger  tagger = UnigramTagger(s)  Daí então, podemos usar o método tag para recuperar os rótulos morfológicos em uma frase qualquer:  frase = "São Paulo é uma cidade grande"  print(tagger.tag(frase.lower().split()))  (a) Encontre algumas frases que contenham palavras que podem assumir classes morfológicas diferentes dependendo do contexto. Como o UnigramTagger lida com essa situação?  (b) O corpus MacMorpho também tem uma base de dados para teste. Abra a base de dados para teste (de maneira similar que fez com a base de treino) e use o método .evaluate() do UnigramTagger para determinar o índice de acertos do tagger na base de teste:  tagger.evaluate(s\_) |

|  |
| --- |
| **Exercício 6** |
| *Objetivo: usar o contexto para determinar classes morfológicas* |
| Considere o trecho:  Segurando a vela, a viúva vela seu finado esposo.  (a) Quais são as classes morfológicas da palavra “vela”?  (b) Como você pode diferenciá-las?  (c) O rotulador por unigramas poderia fazer essa diferenciação?  (d) Teste o rotulador por unigramas nesta frase. O que ele retorna? |

|  |
| --- |
| **Exercício 7** |
| *Objetivo: incorporar contexto na análise morfológica* |
| (a) Como fizemos para incorporar contexto em nosso modelo linguístico da aula anterior?  (b) Analise o código abaixo. O que ele faz? Qual deveria ser a saída deste código?  from nltk.tag import NgramTagger  ngram\_tagger = NgramTagger(1, s, backoff=default\_tagger)  ngram\_tagger.evaluate(s\_)  (c) Qual é o índice de acertos de um rotulador que usa bigramas:  from nltk.tag import NgramTagger  bigram\_tagger = NgramTagger(2, s)  bigram\_tagger.evaluate(s\_)  Como poderíamos melhorar o desempenho dele?  (d) Incorpore trigramas, 4-gramas, etc., à série de rotuladores com backoff. Como o índice de acertos se modifica em relação ao aumento do tamanho do contexto? |

|  |
| --- |
| **Exercício 8** |
| *Objetivo: usar um classificador morfológico em uma ferramenta de suporte ao texto* |
| Um problema que existe na produção de textos é a prolixidade, isto é, quando usamos muitas palavras para passar uma ideia adiante. Há muitas medidas que são indicadores de prolixidade, como por exemplo a densidade léxica. A densidade léxica parte do pressuposto de que os pontos essenciais de um texto estão presentes em verbos e substantivos, e que as outras classes morfológicas contêm somente palavras não-essenciais ao texto e que, portanto, poderiam ser removidas.  Faça uma pequena função que calcula a densidade léxica de uma frase. |

|  |
| --- |
| **Exercício 9** |
| *Objetivo: aplicar PoS tagging para melhorar a qualidade das análises em corpora de textos* |
| NLTK já vem com um PoS Tagger pré-treinado para o inglês:  import nltk  nltk.download('averaged\_perceptron\_tagger')  from nltk import pos\_tag  pos\_tag("this is my testing sentence".split())  Incorpore-o em sua análise do banco de dados IMDB para mostrar *separadamente* quais são os adjetivos, os verbos e os substantivos que mais se relacionam a comentários positivos e negativos. Que tipo de informação podemos tirar de cada uma das classes gramaticais?  Desafio: você conseguiria fazer um sistema que transforma um comentário negativo em um comentário positivo? |