Processamento de Linguagem Natural

Aula 4: Modelos de Linguagem com N-Gramas

Objetivo: ao fim desta aula, o aluno será capaz de entender a relevância de usar n-gramas em análises de texto, e os efeitos de aumentar o tamanho dos n-gramas

|  |
| --- |
| **Exercício 1** |
| *Objetivo: explicar a falha em não considerar a ordem das palavras em textos escritos* |
| Considere os seguintes textos  TEXTO 1:  Ele trabalhava e não faltava comida em casa.  TEXTO 2:  Ele não trabalhava e faltava comida em casa.  (a) Calcule o TF para cada uma das palavras de cada um dos textos.  (b) Se considerarmos somente o TF, há diferenças entre os textos?  (c) O sentido dos textos é o mesmo?  (d) Como sabemos que o sentido dos textos é diferente? |

|  |
| --- |
| **Exercício 2** |
| *Objetivo: relacionar o modelo de unigramas a sua contrapartida probabilística* |
| Quando usamos a medida TF, implicitamente, estamos assumindo um modelo probabilístico para a geração do nosso texto.  (a) Complete a frase abaixo:  *Ao dividirmos a TF de uma palavra w pelo número total de palavras, temos P(\_\_\_) , que é a probabilidade de, ao sortear aleatoriamente \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ dentre o conjunto \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, encontrar \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.*  (b) Proponha uma maneira de gerar novos textos que obedeçam à mesma distribuição de probabilidades de palavras que foi mencionada no item anterior. |

|  |
| --- |
| **Exercício 3** |
| *Objetivo: descrever uma frase em termos de n-gramas* |
| Ao invés de contar palavras isoladas, podemos contar pares de palavras subsequentes, ou bigramas. Por exemplo, na frase:  *“o pato brincou com a bola”*  podemos encontrar os pares:  (o, pato), (pato, brincou), (brincou, com), (com, a), (a, bola)  (a) Encontre os bigramas dos textos do exercício 1.  (b) Ao contar bigramas como se fossem termos (isto é, calcular a frequência de bigramas), podemos diferenciar o conteúdo dos textos?  (c) Faça o mesmo teste, mas agora usando trigramas, e, após, use 4-gramas.  (d) Seria teoricamente possível usar n-gramas, isto é, vizinhanças de algum tamanho inteiro arbitrário, para extrair o conteúdo do texto? |

|  |
| --- |
| **Exercício 4** |
| *Objetivo: observar o efeito de usar n-gramas na análise de conjuntos de texto* |
| O CountVectorizer do scikit-learn tem suporte a calcular n-gramas de diferentes tamanhos. A sintaxe para isso é algo como:  vectorizer = CountVectorizer(ngram\_range=(1,2))  os números 1 e 2 devem ser trocados pelo mínimo e máximo n dos n-gramas que serão considerados na contagem.  Usando sklearn, calcule o Document Frequency dos bigramas e trigramas na coleção IMDB (pode usar o notebook da aula anterior para isso)  (a) Quais são os bigramas e trigramas mais comuns entre os reviews positivos? E entre os negativos?  (b) Essa análise trouxe algum insight novo em relação à anterior, feita somente com os unigramas? |

|  |
| --- |
| **Exercício 5** |
| *Objetivo: relacionar o DF esperado de um n-grama ao número n* |
| Um n-grama é uma sequência de tokens com n tokens específicos consecutivos.  (a) Numa coleção, estamos calculando o Document Frequency (DF) – isto é, a quantidade de documentos – em que cada n-grama aparece. Como você esperaria que os valores de DF se comportassem com o aumento do número N? Em outras palavras, o que acontece com DF para n-gramas cada vez maiores?  (b) Use a base de dados IMDB para confirmar a sua hipótese.  (c) Em uma coleção de textos, em que situação teríamos um DF diferente de 1 para um n-grama de tamanho 30?  (d) Qual seria uma possível aplicação para esse conceito? |

|  |
| --- |
| **Exercício 6** |
| *Objetivo: relacionar bigramas ao modelo probabilístico correspondente* |
| Considere o texto:  *Não fumo, não bebo. Eu corro, eu treino.*  (a) Calcule o TF dos bigramas deste texto.  (b) Calcule a probabilidade de, ao sortear um bigrama aleatório do texto, encontrar um bigrama específico (escolha o bigrama com que você quer trabalhar).  (c) Como poderíamos gerar um novo texto de tal forma a preservar a probabilidade dos bigramas?  (d) Complete a frase abaixo:  A probabilidade P(w(t)|w(t-1)) é igual ao número de vezes que a palavra w aparece \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ dividido pelo número total de \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.  (e) Como poderíamos realizar o mesmo processo, mas usando n-gramas ao invés de bigramas? |

|  |
| --- |
| **Exercício 7** |
| *Objetivo: explicar os passos para treinamento de modelos linguísticos na API do NLTK* |
| Tome por base o notebook “exemplo de treinamento de modelo linguístico”. Execute-o e, tomando por base as saídas (e outras saídas que você crie), busque responder:  (a) no **passo 1**, como é a estrutura de listas que permite representar frases e palavras de uma coleção?  (b) no **passo 2**, o que faz a função everygrams()? E a função pad\_both\_ends()? Por que é necessário usar pad\_both\_ends()?  (c) no **passo 3**, o que faz a função flatten()?  (d) no **passo 4**, quais informações são necessárias para fazer o fit() do modelo?  (e) no **passo 5**, encontre a probabilidade de a palavra “nada” seguir a palavra “frase”.  (f) no **passo 6**, certifique-se de que o modelo gerado com a API mais condensada é equivalente ao gerado anteriormente.  (g) no **passo 7**, modifique o random seed e o número de palavras a gerar para gerar diferentes frases. O que acontece com as frases geradas se o modelo usar apenas unigramas (n=1)? |

|  |
| --- |
| **Exercício 8** |
| *Objetivo: programar um gerador de conteúdo textual* |
| Tomando por base os códigos que você já tem (desta e das aulas anteriores), faça um programa que gere textos que poderiam se passar por comentários sobre filmes deixados em redes sociais (dica: use a base IMDB!)  Alterando o tamanho dos n-gramas usados, observe:  (a) Como os textos gerados se tornam mais ou menos realísticos?  (b) Como o tempo de treinamento do modelo se altera?  (c) Como a variabilidade dos textos gerados se altera (ou: quanto do material gerado parece uma cópia dos textos usados como fonte?) |

|  |
| --- |
| **Mini-projeto** |
| *Objetivo: analisar os problemas ligados à detecção automática de plágio* |
| Neste mini-projeto, faremos uma prova de conceito de um detector automático de plágio em textos escritos. Esse tipo de programa serve para detectar automaticamente se as respostas dadas por alunos em provas discursivas são textos originais ou se são cópias.  a) Faça (e execute!) um plano para agregar uma base de dados que permita fazer um teste do detector de plágio. Dica: pode ser interessante fazer isso com a turma toda colaborando!  b) Programe o detector de plágio. Ele funciona? Quais casos de teste você testou?  c) Tente quebrar seu detector de plágio! Como você conseguiria burlar as regras que você mesmo inventou? Nesta etapa, calibre parâmetros do detector para que ele funcione tão bem quanto possível (por exemplo: ajustar o tamanho dos n-gramas, funcionar com maiúsculas e minúsculas, etc.).  d) Após ter calibrado seu detector de plágio, encontre um(a) colega que também já tenha terminado a calibração. Tente burlar o detector de plágio dele(a). Você encontrou diferenças em relação ao seu?  e) Que casos de plágio já somos capazes de detectar? Que casos ainda não conseguimos? |