Universidade Federal do Rio Grande do Sul - Instituto de Informática
Departamento de Informática Aplicada - Processamento de Linguagem Natural
Prof. Dennis Giovani Balreira

Lista de Exercícios - Módulo 1 (Respostas)

1. Processamento de Linguagem Natural (PLN) é o campo da inteligência artificial que busca a interação entre máquinas e linguagem humana, permitindo que computadores analisem, compreendam e gerem texto ou fala.

2. Compreensão de linguagem natural (NLU) envolve a interpretação de significado e intenção do texto. Exemplos: análise de sentimento, reconhecimento de entidades nomeadas. Geração de linguagem natural (NLG) refere-se à criação de texto em linguagem humana. Exemplos: geração de texto e respostas a perguntas.

3. Não podemos afirmar que "passamos no teste de Turing" porque enganar um humano não prova que a máquina possui compreensão ou inteligência genuína, apenas que ela pode imitar comportamentos humanos.

4.
- Modelos baseados em regras (1950-1984): Utilizam regras linguísticas programadas manualmente.
- Modelos estatísticos (1985-2012): Aplicam técnicas estatísticas para processar linguagem.
- Word embeddings fixas + Deep learning (2013-2017): Representações numéricas fixas para palavras em redes neurais.
- Word embeddings contextuais + Transformer - NLU (2018-2022): Modelos como BERT que entendem linguagem considerando o contexto.
- Word embeddings contextuais + Transformer - NLG (2023-?): Modelos como GPT que geram texto de alta qualidade com base no contexto.

5. Modelos baseados em regras utilizam conjuntos de regras linguísticas pré-definidas. Desvantagens: não são escaláveis para grandes volumes de dados e são difíceis de adaptar a novos domínios.

6. Dois fatores que impulsionaram o crescimento de PLN são:
- A disponibilidade de grandes volumes de dados para treinar modelos robustos.
- O avanço em hardware, como GPUs e TPUs, que tornam o treinamento mais eficiente.

7. Dado é um valor bruto e descontextualizado. Informação é o dado processado e contextualizado, gerando significado.

8.
- Dados estruturados: Organizados em tabelas com formato fixo.
- Dados semi-estruturados: Possuem alguma organização, como XML e JSON.
- Dados não-estruturados: Não seguem uma organização definida, como texto livre.

Os dados não-estruturados são os mais abundantes e mais difíceis de processar.

9. A semântica apresenta maior desafio porque exige compreensão de significados e contextos complexos, algo difícil para máquinas.

10. A Lei de Zipf afirma que a frequência de uma palavra é inversamente proporcional à sua posição no ranking de frequências. Stopwords são frequentes, estando muito provavelmente no topo do ranking.

11. A Lei de Heaps diz que o vocabulário cresce sublinearmente com o tamanho do texto. Em geral, a taxa de novos termos diminui conforme o texto aumenta.

12. Modelos probabilísticos baseiam-se em probabilidades condicionais, prevendo a próxima palavra com base no “contexto” fornecido (palavras anteriores).

13. Modelos N-gramas são uma forma de modelo probabilístico que considera apenas as últimas n palavras do contexto. A diferença para o caso geral é que este pode utilizar contextos mais amplos.

14. Modelos N-gramas com valores pequenos de n não capturam dependências de longo alcance, como as entre "bolo" e "delicioso" no exemplo.

15. Expressões regulares são padrões utilizados para buscar e manipular texto. São importantes no pré-processamento para remover ruído e estruturar os dados textuais.

16. Não é correto afirmar que todo texto deve passar por todas as técnicas de pré-processamento antes de serem utilizados, pois as técnicas devem ser escolhidas conforme o propósito da tarefa. Por exemplo, a remoção de stopwords pode não ser adequada para tarefas que dependem de preposições ou palavras de ligação.

17. A estratégia comum para segmentação de sentenças em línguas ocidentais como o inglês e português é o uso de delimitadores como pontos finais e vírgulas. Essa abordagem pode não funcionar bem para línguas como o chinês e japonês, que não possuem espaços claros para separar palavras e sentenças.

18. A remoção de stopwords pode atrapalhar tarefas de classificação de sentimentos, pois palavras como "não" são fundamentais para determinar polaridade (ex.: "não gostei").

19. Utilizar stemming em palavras como “bebê” e “bebendo” pode levar a resultados inadequados, como a redução de ambas ao mesmo radical (“beb”), o que distorce o significado distinto entre substantivo e verbo.

20. A distância de Levenshtein entre “gato” e “ratos” considerando os custos (1,1,2) é 3:

- Substituir 'g' por 'r' (custo 2).

- Inserir 's' no final (custo 1).

21. Representar caracteres por números ASCII pode introduzir relações espúrias entre os valores numéricos, como associar significado a distâncias entre códigos que não têm relação semântica.

22. O modelo Bag of Words (BoW) representa documentos como vetores, onde cada posição corresponde à frequência de uma palavra do vocabulário. O TF-IDF mitiga o problema de dar peso excessivo a palavras muito frequentes, como stopwords.

23. TF (Term Frequency) mede a frequência de um termo em um documento. IDF (Inverse Document Frequency) avalia a raridade do termo em todos os documentos. O IDF está relacionado à raridade do termo.

24. Para o conjunto de documentos:
d1: "gato e cachorro"
d2: "gato gosta de leite"
d3: "cachorro gosta de osso"

a) Vetor do vocabulário: ["gato", "e", "cachorro", "gosta", "de", "leite", "osso"]

b) Matriz BoW:

	gato	e	cachorro	gosta	de	leite	osso
d1	1	1	1	0	0	0	0
d2	1	0	0	1	1	1	0
d3	0	0	1	1	1	0	1

c) Matriz TF-IDF (em log):
TF-IDF = TF * log(N/DF), onde N = 3 (total de documentos) e DF é o número de documentos contendo o termo.

	gato	e	cachorro	gosta	de	leite	osso
d1	1 * log(3/2)	1 * log(3/1)	1 * log(3/2)	0	0	0	0
d2	1 * log(3/2)	0	0	1 * log(3/2)	1 * log(3/2)	1 * log(3/1)	0
d3	0	0	1 * log(3/2)	1 * log(3/2)	1 * log(3/2)	0	1 * log(3/1)

25. Métodos baseados em distância de edição, como Levenshtein, medem a similaridade entre textos com base no número de operações necessárias para transformar um texto em outro (inserções, deleções, substituições). Já medidas de similaridade vetorial, como a distância de cosseno, comparam textos representados como vetores, considerando o ângulo entre eles. Distância de edição é mais apropriada para pequenas sequências de texto, como palavras ou strings curtas, enquanto a distância de cosseno é mais adequada para comparar documentos maiores e representações vetoriais, como BoW ou embeddings.

26. Mesmo utilizando a distância de cosseno, BoW e TF-IDF não capturam bem a similaridade semântica porque não consideram o contexto ou as relações entre palavras. Embeddings fixas, como word2vec ou GloVe, conseguem representar palavras em um espaço contínuo que reflete semelhanças semânticas, resultando em uma representação mais rica.

27. A construção de datasets para aprendizado supervisionado é custosa porque requer anotações manuais feitas por especialistas, além de garantir um balanceamento e diversidade suficientes para treinar modelos robustos. Isso demanda tempo, conhecimento e recursos humanos.

28. Para aplicar BoW sobre letras de músicas, os seguintes passos poderiam ser feitos:

(i) Tokenização: Essencial para segmentar o texto em palavras ou frases.

(ii) Remoção de pontuação: Relevante se a pontuação não agregar significado. Porém, em letras de músicas, pode haver casos onde pontuação contribui para a interpretação.

(iii) Case folding: Útil para evitar distinções desnecessárias entre maiúsculas e minúsculas.

(iv) Remoção de stopwords: Depende do objetivo. Stopwords podem ser importantes em letras de músicas, já que conectam ideias e podem afetar o significado.

29. As probabilidades no algoritmo Naive Bayes não são exatas porque ele assume independência condicional entre os atributos, o que raramente ocorre no texto, já que palavras têm dependências. Essa falta de exatidão não é um problema para aprendizado de máquina porque o objetivo é otimizar a separação entre classes, e não modelar a distribuição estatística com precisão absoluta.

30. Generalização em modelos de aprendizado de máquina é a capacidade do modelo de realizar boas predições em dados não vistos, além dos dados usados no treinamento. Para garantir que o modelo está generalizando corretamente, é necessário avaliar seu desempenho em um conjunto de teste separado e utilizar validações cruzadas para confirmar a consistência.

31. Na divisão com holdout, os dados são separados em conjuntos de treino e teste, geralmente em proporções como 70/30 ou 80/20. A divisão estratificada preserva a proporção de classes no conjunto original, garantindo que ambos os conjuntos reflitam a distribuição dos dados, o que é essencial em problemas desbalanceados.

32. A abordagem k-fold cross validation é indicada porque utiliza todas as partes do dataset tanto para treino quanto para teste, reduzindo a variabilidade associada a uma única divisão dos dados, como ocorre no holdout.

33. O conjunto de validação é usado para ajustar hiperparâmetros e evitar o sobreajuste ao conjunto de treino. Ele ajuda a selecionar o melhor modelo antes de testá-lo no conjunto de teste final.

34.

a) A imagem (A) representa underfitting, enquanto a imagem (B) representa overfitting.

b) No underfitting, o modelo é muito simples e não consegue capturar a relação entre os dados. No overfitting, o modelo é excessivamente complexo e ajusta-se demais aos dados de treino, capturando até o ruído.

c) A curva ideal seria uma que segue a tendência geral dos dados sem ajustá-los de forma rígida ou ignorá-los completamente, alcançando um equilíbrio entre simplicidade e ajuste.

35. O F1-Score é utilizado porque é uma métrica que balanceia precisão e revocação, sendo mais apropriado em datasets desbalanceados, onde a acurácia pode ser enganosa ao favorecer a classe majoritária.

36. Representações tradicionais como BoW e TF-IDF não capturam contexto, tratam palavras como independentes e geram vetores esparsos de alta dimensionalidade.

37. A hipótese distribucional afirma que palavras com significados semelhantes aparecem em contextos semelhantes. Por exemplo, "cão" e "cachorro" frequentemente aparecem em contextos relacionados a animais domésticos.

38. Não é possível definir completamente o significado de uma palavra sem seu contexto, pois o uso e o significado variam dependendo da situação em que a palavra é empregada.

39. Word embeddings fixas são vetores densos que mapeiam palavras em espaços contínuos, capturando semelhanças semânticas. Elas diferem de BoW e TF-IDF porque não dependem de frequências e consideram relações entre palavras. São mais avançadas porque capturam semântica e reduzem a dimensionalidade.

40. As word embeddings fixas não conseguem diferenciar significados contextuais das palavras (ex.: "banco" como instituição financeira e "banco" como assento). Isso pode impactar negativamente tarefas em PLN que exigem desambiguação semântica ou adaptação a contextos variados.

41. O treinamento de embeddings como Word2Vec utiliza grandes corpora de texto para identificar padrões de coocorrência entre palavras em janelas de contexto. Um corpus maior e mais diverso resulta em embeddings de maior qualidade, pois proporciona uma melhor representação de palavras em diferentes contextos e usos.

42. No algoritmo Word2Vec no modo skip-gram, o modelo tenta prever as palavras de contexto dado o termo central. Após o treinamento, os vetores da camada de entrada (ou intermediária) são usados como as embeddings finais para cada palavra.

43. Embeddings fixas possuem vieses porque refletem os padrões do corpus com que foram treinadas. Se o corpus contém estereótipos ou preconceitos, essas características são incorporadas às embeddings.

44. Embeddings fixas são limitadas porque atribuem a mesma representação vetorial a uma palavra, independentemente de seu contexto. Embeddings contextuais, como BERT, superam isso gerando representações diferentes para a mesma palavra dependendo do contexto em que ela aparece.

45. Modelos de redes neurais são estruturas computacionais inspiradas no cérebro humano, compostas por camadas de nós interconectados. Aplicados a textos, podem ser usados para tarefas como classificação, geração de texto e tradução automática, com representações vetoriais como entrada.

46.

a) Perceptrons são as unidades básicas de redes neurais, com pesos ajustáveis para cada entrada e um bias que define deslocamentos.

b) Camadas de entrada recebem os dados iniciais, camadas ocultas processam a informação e camadas de saída fornecem os resultados.

c) A função de ativação adiciona não-linearidade, permitindo que o modelo capture padrões complexos (ex.: ReLU, Sigmoid).

d) A função de perda mede o erro entre a saída prevista e o valor esperado, guiando o treinamento (ex.: entropia cruzada).

e) Gradiente descendente é o método de otimização usado para ajustar os pesos da rede com base no gradiente da função de perda.

f) Backpropagation é o algoritmo usado para calcular o gradiente dos pesos, propagando os erros de volta na rede.

g) Deep learning refere-se a redes neurais profundas, com muitas camadas, capazes de modelar dados complexos e de alta dimensionalidade.

$a_1^{(1)}$: 0.30

$a_2^{(1)}$: 0.30

$\hat{y}$: 0

47. Erro (MAE): 1.5

48.

a) RNN (Recurrent Neural Networks): Redes neurais que processam sequências de dados ao permitir conexões recorrentes, utilizando o estado anterior para informar o processamento atual.

b) LSTM (Long Short-Term Memory): Variante das RNNs projetada para capturar dependências de longo prazo, mitigando problemas de desaparecimento e explosão de gradientes.

c) Bi-LSTM: LSTMs bidirecionais que processam a sequência em ambas as direções (frente e verso), permitindo capturar melhor o contexto global.

d) Encoder-decoder: Arquitetura composta por duas partes: o encoder codifica a entrada para uma representação intermediária e o decoder gera a saída a partir dessa representação.

e) Attention e self-attention: Mecanismos que destacam as partes mais relevantes do contexto. Self-attention foca nas interações entre palavras na mesma sequência.

f) Transformers: Arquitetura que substitui recorrência por atenção, permitindo processamento paralelo e captura eficiente de dependências contextuais.

49.

a) Desaparecimento e explosão de gradientes: Os gradientes se tornam muito pequenos ou muito grandes em sequências longas. Solução: usar LSTM ou GRU.

b) Dificuldade em aprender dependências de longo prazo: RNNs simples esquecem informações antigas. Solução: LSTMs e mecanismos de atenção.

c) Treinamento lento (sequencial): As RNNs processam dados em sequência. Solução: usar Transformers para processamento paralelo.

50.

a) Encoder-decoder: T5, usado para tradução e sumarização.

b) Encoder-only: BERT, usado para classificação e análise de sentimentos.

c) Decoder-only: GPT, usado para geração de texto.

51. O mecanismo de atenção identifica as partes mais importantes do contexto para cada palavra, atribuindo pesos que refletem sua relevância. No self-attention, a relação entre todas as palavras de uma sequência é considerada.

52. Transformers permitem treinamento em grandes volumes de dados porque processam sequências paralelamente, enquanto RNNs e LSTMs dependem de processamento sequencial.

53. Word embeddings contextuais lidam com polissemia ao gerar representações únicas para cada palavra baseada no contexto. Diferentemente de embeddings fixas, uma palavra como "banco" terá representações distintas dependendo do uso. Exemplo: "banco de madeira" vs. "banco financeiro".

54. Sim, todas LLMs conhecidas atualmente utilizam a arquitetura Transformer como base, devido à sua eficiência no processamento paralelo e captura de dependências contextuais.

55.

a) BERT:

(1) Coleta de textos variados;

(2) Tokenização com WordPiece;
(3) Pré-treino com Masked Language Modeling (MLM) e Next Sentence Prediction (NSP);
(4) Fine-tuning para tarefas específicas.

b) GPT-3:
(1) Coleta massiva de dados;
(2) Tokenização com Byte Pair Encoding (BPE);
(3) Pré-treino autorregressivo;
(4) Ajustes com prompts para aplicações específicas.

56.
(1) Tradução automática: Encoder-decoder.
(2) Classificação: Encoder.
(3) Análise de sentimentos: Encoder.
(4) Geração de texto: Decoder.
(5) Sumarização: Encoder-decoder.

57. BERT e GPT são treinados de forma auto-supervisionada porque utilizam os próprios dados para gerar rótulos (ex.: prever palavras ou completar sentenças) sem necessidade de anotações manuais.

58. Estratégias como BPE e WordPiece geram subpalavras, permitindo melhor representação de termos raros e reduzindo o vocabulário. São mais eficientes do que segmentar apenas por espaços.

59.
a) Masked Language Modeling (MLM): Algumas palavras são mascaradas e o modelo tenta prevê-las.
b) Next Sentence Prediction (NSP): O modelo aprende se uma frase segue logicamente a anterior.

60.
Após o pré-treino, o BERT fornece embeddings contextuais para cada token. Para tarefas específicas, como classificação, camadas adicionais são adicionadas para ajustar as saídas ao objetivo.

61.
a) Pré-treino: Treinamento inicial do modelo em tarefas gerais.
b) Continuação de pré-treino: Ajustes adicionais usando dados do domínio específico.
c) Fine-tuning: Ajuste final para uma tarefa específica com rótulos.

62.
a) BERT é bidirecional, o que o torna menos adequado para modelos autorregressivos que geram texto palavra a palavra.

b) Ele é treinado para entender o texto, não para geração sequencial.

63. Modelos autorregressivos, como GPT, preveem a próxima palavra com base nas anteriores. Estratégias de decodificação, como beam search ou sampling, controlam a geração do texto.

64. GPT é unidirecional porque utiliza apenas o contexto anterior para prever a próxima palavra. No "Masked Self Attention", partes futuras são mascaradas para evitar que influenciem a previsão.

65. Abordagens como prompts ou adaptação leve são úteis para LLMs porque evitam re-treinar o modelo inteiro, economizando recursos e tempo.

66. Zero-shot: O modelo realiza tarefas sem exemplos específicos de treinamento. Few-shot: O modelo utiliza poucos exemplos para ajustar a tarefa.

67.
a) Papagaios estocásticos: Repetem padrões sem entendimento.
b) Halucinações: Geram informações incorretas ou irreais.
c) Raciocínio: Falham em realizar deduções complexas ou lógicas.