## Capítulo 4 - Agentes de Aprendizagem

1. Para o vosso componente RAG, o sistema precisa de classificar os documentos recuperados como “altamente relevantes” ou “pouco relevantes”. Proponha uma abordagem de aprendizagem supervisionada para treinar este classificador. Que atributos (features) dos documentos poderiam ser usados?

1. Criação do conjunto de treino:
   * Coleta de documentos já classificados por especialistas, como “altamente relevante” ou “pouco relevante”
2. Extração de features:
   * Converter cada documento num vetor de características numéricas.
3. Treino do modelo:
   * Aplicar algoritmo supervisionado (como SVM, k-NN, ou redes neuronais com back propagation), aprender a fronteiro entre os dois tipos de relevância.
4. Validação e ajuste:
   * Avaliar desempenho (ex: F1-score) e ajustar hiperparâmetros como o k (no k - NN) ou C e y (no SVM)

Features possíveis:

* Similaridade com a consulta
* Presença de termos-chave
* Embeddings semânticos
* Comprimento
* Posição dos termos

2. Imagine que quer usar uma Árvore de Decisão para o problema do exercício anterior. Descreva como o algoritmo construiria o primeiro nó da árvore.

O algoritmo analisa todas as features disponíveis nos documentos para escolher a melhor que separa as classes

A escolha é feita por um grau de pureza do algoritmo de acordo de quanto cada feature consegue distinguir documentos dos mais relevantes aos menos relevantes, recorrendo a métricas como ganho de informação (com base na entropia) ou por índice Gini e pela feature que apresentar maior ganho de informação fica como primeiro nó da árvore de decisão.

Exemplo:

Se entre as features tivermos:

* Similaridade com a pergunta
* Autoridade da Fonte
* Comprimento do texto

E o algoritmo descobrir que a similaridade com a pergunta separa melhor os documentos entre duas classes, então essa feature será o primeiro nó da árvore de decisão

3. O agente “Escritor” pode ser melhorado para adaptar o seu estilo de escrita (formal vs. informal) com base no pedido do utilizador. Como poderia uma rede neuronal simples ser treinada para esta tarefa de classificação? Defina as entradas e as saídas.

Entradas : representam caracteristicas extraídas pelo pedido do utilizador ou do texto:

* Vetores de palavras
* Presença de expressões típicas
* Estrutura sintática
* Indicação direta do utilizador

Saída: uma variável binária:

* 1 - estilo formal
* 0 - estilo informal

4. No vosso projeto, o sistema precisa de agrupar artigos de investigação semelhantes para análise. Que tipo de paradigma de aprendizagem (supervisionado, não supervisionado, por reforço) é mais adequado para esta tarefa de clustering? Justifique.

O paradigma mais adequado será a aprendizagem não supervisionada, pois o objetivo será agrupar automaticamente artigos semelhantes. O sistema analisa as características dos artigos e forma grupos com base na sua semelhança interna. Ao contrário da aprendizagem supervisionada, o sistema irá descobrir padrões e relações entre os dados por si próprio.

5. Compare a adequação de k-NN e SVM para classificar em tempo real se a resposta de um agente é “segura” ou “não segura” (e.g., contém informação sensível). Considere a velocidade de previsão e a interpretabilidade.

Para classificar algo em tempo real o mais adequado é o SVM, pois oferece previsões rápidas após o treino. Ao contrário do k-NN que apesar de ser mais intuitivo é mais lento, portanto é menos indicado para aplicações em tempo real.

6. O “Gerador de Problemas” é um componente de um agente de aprendizagem. No contexto do “Cogni - Verse”, como poderia um gerador de problemas sugerir novas tarefas de investigação para o sistema treinar e melhorar autonomamente?

O gerador de problemas pode sugerir novas tarefas para que o sistema aprenda melhor e melhor sozinho.

Por exemplo:

* Analisar lacunas no conhecimento atual
* Sugerir novos tópicos de pesquisa
* Criar variações de tarefas anteriores para testar diferentes abordagens
* Propor experiências de comparação entre agentes

7. O vosso sistema usa o smol-agents, que gera código Python. Como poderia um agente de aprendizagem ser treinado para detetar e corrigir erros comuns no código gerado por outros agentes?

Um agente de aprendizagem pode ser treinado para detetar e corrigir erros comuns no código ao :

* Aprender com exemplos ou então recompensas
* Integrar modulos de critica e autoavaliação
* Usar técnicas simbólicas (regras) como conexionistas (redes neuronais)
* Adaptando continuamente, vai fazer com que se torne um agente de aprendizagem adaptativo capaz de melhorar a qualidade do codigo de forma autónoma