Mauricio Ferreira de Araujo - 222007021
Disciplina: Inteligência Artificial

UnB

Portfólio 2 - Projeto de busca informada

1 Introdução

Este documento tem como foco apresentar o problema a ser resolvido, a implementação e os resultados de um algoritmo de Busca informada, desenvolvido na linguagem Python, para a disciplina de Inteligência Artificial.

Foram utilizados a IDE pycharm para implementar o algoritmo de Beam Search.

2 Problema

Imagine que você precisa tomar uma série de decisões para chegar a um objetivo. O problema é que cada decisão que você toma abre um novo leque de opções, e cada uma dessas opções abre outro leque, e assim por diante. Se você tentar explorar todas as combinações possíveis para garantir que encontrou a melhor com brute force, o número de caminhos cresce exponencialmente. Para qualquer problema do mundo real, isso é computacionalmente impossível.

O beam search resolve esse problema porque ele é uma forma inteligente de explorar vários caminhos promissores ao mesmo tempo superando uma busca gulosa, mas sem explorar todos os caminhos como um brute force.

Um problema comum é a tradução automatica pois com frases ambiguas traduções gulosas podem inverter o sentido das frases, o beam search soluciona esse problema segurando não só a melhor possibilidade local mas as k possiveis e ao segurar as k melhores sequências (hipóteses) a cada passo, ele dá chance para que uma tradução que começou parecendo "um pouco pior"possa provar seu valor se tornando a melhor tradução global no final.

3 Algoritmo desenvolvido de busca informada: Beam search

O algoritmo beam search inicia com a definição do tamanho do feixe (k), por exemplo, $k = 3$. A busca começa, e a cada iteração, o algoritmo mantém as k (ou seja, 3) hipóteses mais prováveis encontradas até o momento. O passo crucial é que o algoritmo expande todas essas 3 hipóteses, gerando um novo conjunto de múltiplos candidatos. Deste novo conjunto, ele seleciona novamente, mantendo apenas os 3 melhores no geral. Este ciclo se repete sucessivamente até que uma condição de parada, como o fim da sequência (como no exemplo da tradução automatica) ou um limite de tamanho, seja alcançada.

3.1 Exemplo de uso

A ideia de execução definida no código era a de montar um sistema que simula um modelo de tradução usando tokens e a probabilidade de cada um dos tokens de ser o

melhor. A geração da probabilidade foi estipulada previamente em um json. Dessa forma, ao executar, o algoritmo faz a comparação com uma busca gulosa que pega somente o melhor com o beam search que armazena os k melhores, mostrando a tradução esperada em um modelo de tradução como mostrado na Figura 1.

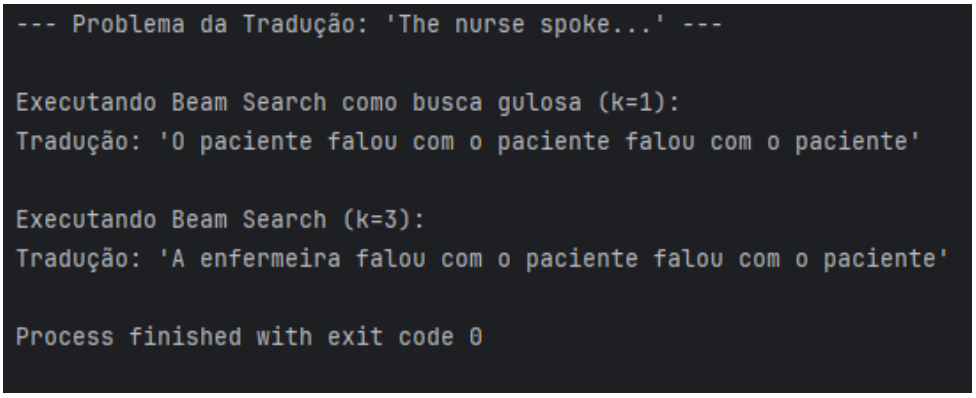

Figura 1: Execução do beam search