

Course: Introduction to Artificial Intelligence

Instructor: Dr. Syed Ali Raza

Spring 2026

Institute of Business Administration, Karachi

Assignment #1

Assignment Title

Word Ladder Search in Semantic Embedding Space

Note: read every instruction very carefully.

Learning Objectives

After completing this assignment, students will be able to:

1. Formulate a real-world-inspired problem as a **state-space search problem**.
 2. Implement and compare **uninformed search algorithms** (BFS, DFS, UCS).
 3. Implement and compare **informed search algorithms** (Greedy Best-First Search, A*).
 4. Design and analyze **heuristic functions** based on vector similarity.
 5. Empirically evaluate search strategies in a **large, implicit state space**.
 6. Critically analyze the strengths and limitations of heuristic search.
-

Problem Description

In classical AI, a *word ladder* problem transforms one word into another through a sequence of valid intermediate words. In this assignment, you will generalize this idea using **word embeddings**.

Each word is represented as a point in a continuous embedding space. Your task is to design a search environment in which:

- Each **state** corresponds to a word.
- An **action** moves from the current word to one of its nearby words in embedding space.
- The objective is to reach a **goal word** starting from a **source word**, using search algorithms.

You will explore how uninformed and informed search algorithms behave in this semantic space and evaluate the effectiveness of a heuristic. You may use cosine similarity or a closely related similarity-based heuristic. Any alternative must be clearly defined and justified.

You are given a **pretrained word embedding file** for GloVe 100-dimensional word embeddings (glove.100d.20000.txt). It is a **restricted vocabulary list** (20,000 words). Read more about GloVe here: <https://nlp.stanford.edu/projects/glove/>

No starter code is provided. You must design and implement the solution independently.

Task 1: Problem Formulation (20%)

Clearly define the following components in your report:

1. **State Representation**
 - What constitutes a state?
 2. **Initial State and Goal Test**
 - How are the start and goal words defined?
 3. **Action Space**
 - How do you define neighboring words?
 - How many neighbors (k) are considered at each step?
 - Choose k such that $5 \leq k \leq 50$. You must justify your chosen value.
 4. **Path Cost Function**
 - Uniform cost or similarity-based cost?
 - Justify your choice.
 5. **Heuristic Function**
 - Define the heuristic mathematically.
 - Explain its intuition.
-

Task 2: Search Algorithm Implementation (30%)

Implement the following algorithms:

Uninformed Search

- Breadth-First Search (BFS)
- Depth-First Search (DFS)
- Uniform Cost Search (UCS) (Think about its cost function)

Informed Search

- Greedy Best-First Search

- A* Search

Your implementation must construct the search graph **implicitly** (neighbors computed during expansion). Avoid revisiting already explored states. Terminate when the goal word is reached or when no solution exists.

For DFS, you may use a depth limit or other reasonable safeguards to ensure termination.

Task 3: Experimental Evaluation (25%)

Pick at least **five (start, goal) word pairs**.

For each pair and each algorithm, report:

- Whether a path was found
- Path length (number of steps)
- Number of nodes expanded
- Total runtime

At least one pair should be semantically distant. This forces failure cases.

Present results in a table and discuss observed differences.

Task 4: Analysis and Discussion (15%)

Answer the following questions:

1. Compare uninformed and informed search in terms of efficiency.
 2. Provide an example where Greedy Best-First Search fails to find a good path.
 3. Discuss whether cosine similarity is an admissible and/or consistent heuristic.
 4. Identify at least one case where semantic similarity is misleading.
 5. How does the choice of k (number of neighbors) affect performance?
-

Task 5: GUI (10%)

The GUI should allow a user to interactively explore word-ladder search in the embedding space and observe differences between search algorithms.

Your GUI must support the following:

1. **Input Controls**
 - Text input for:
 - Start word

- Goal word
 - Dropdown or radio button to select the search algorithm:
 - BFS
 - DFS
 - Greedy Best-First Search
 - A* Search
 - (Optional) Control to adjust the number of neighbors (k)
- 2. **Execution**
 - A button to execute the selected search algorithm.
 - The search must run on-demand based on user input.
- 3. **Output Display**

Display the following results clearly:

 - Whether a path was found
 - The resulting word ladder (sequence of words)
 - Length of the ladder (number of steps)
 - Number of nodes expanded
 - Time taken to complete the search
- 4. **Usability**
 - The interface does not need to be visually sophisticated.
 - Clarity, correctness, and interactivity are more important than aesthetics.
 - If the start or goal word is not in the vocabulary, your system should report this clearly and terminate gracefully.

Recommended Tools

You may use:

- **Streamlit** (recommended), or
- Any lightweight GUI or web-based framework of your choice.

Evaluation Note

The GUI is not evaluated on design quality but on:

- Correct integration with your search algorithms
 - Accurate reporting of results
 - Ability to reproduce experimental findings interactively
-

Application Perspective (Conceptual)

Briefly discuss how semantic search paths could be useful in a real-world domain such as medicine or information retrieval. This discussion is **conceptual only**; no domain validation is required.

Submission Guidelines

Submit a single ZIP file containing:

1. Source code
 2. A PDF report (6–8 pages) including:
 - Problem formulation
 - Algorithm descriptions
 - Experimental results
 - Analysis and discussion
-

Academic Integrity

- You may use standard libraries for numerical computation.
- You may consult external resources for understanding algorithms.
- You must **not** copy code from AI assistants or other sources.
- You must clearly understand and be able to explain every part of your submission.

Submissions may include follow-up oral questioning or code walkthroughs.

Grading Rubric (Summary)

Component	Weight
Problem formulation	20%
Correctness of implementation	30%
GUI implementation	10%
Experimental evaluation	25%
Analysis and insight	15%

GUI marks are awarded separately and do not overlap with core algorithm correctness.
