# The Wikipedia Game

Mark Kong, Benjamin Rafetto, and Claire Stolz

CS 182 --- Final Project --- Fall 2018

The Wikipedia Game is a path-finding challenge where a player is presented with two Wikipedia pages and is asked to find a path from the start page to the goal page by clicking links to travel from article to article. The player's score is determined by the length of the path and the time it took to find the path. Due to the interconnectedness of various topics and moderately large branching factor, there is a wide variety of possible paths. Our goal is to solve this game in as efficient a manner as possible.

## Introduction

Given two Wikipedia pages the problem is to find a path between the two. Each page is a state, and each link to a page is a directed edge of cost 1.

For example, given **Sherlock Holmes** → **Magnetic resonance imaging**, one path is Sherlock Holmes → Benedict Cumberbatch → Doctor Strange (2016 film) → Neurosurgery → Magnetic resonance imaging**.**

Rewards are determined by the length of the path and the number of nodes expanded. Here, the number of nodes expanded is a proxy for the amount of time taken to find the path. Technically the Wikipedia game also forbids backtracking, meaning humans must use a strictly greedy approach. We develop algorithms for solving the classic game (e.g. playing as a human), and a game where backtracking is allowed.

## Data

Naturally, our data come from Wikipedia. Presently, we use the the `pywikibot` library to live-query the Wikipedia API. This library gives us access to a large amount of information about each page, but the most important features are the page text, the ingoing links, and the outgoing links for each page. We only use pages in English.

We also use `word2vec`, which models the "closeness" of words. The primary `word2vec` model is a two-layer neural network trained on a large dataset of text, or 'corpus.' Initially, we use the `word2vec` package trained on a set of Google news articles.

The results section that follows uses a set of 50 randomly chosen word pairs to evaluate each algorithm.

## Approach

We consider the following algorithmic approaches:

1. Breadth first search ✯
2. Bi-directional breadth first search ✯
3. Greedy search using Google's `word2vec` package.
    1. Avg., min, and combined distance measure based on words
    2. Avg., min, and combined distance measure based on phrases
    3. Trained on full text of linked page ✯
    4. Additional `word2vec` training on Wikipedia corpus
4. A* search ✯
    1. Heuristic based on text only (as above, in Greedy search)
    2. Heuristic based on branching factor
    3. Compound heuristic using 1 & 2, with a possibility of learning heuristic feature weights
    4. Clustering based on interconnectedness

✯ indicates that the algorithm uses data or techniques not available to the typical human player.

## Results

We present our preliminary results for Greedy search using Google's `word2vec` package, for `word2vec` distance metrics based on (1) words only, and (2) phrases. For each heuristic, we also develop 3 sub-heuristics based on the closest `word2vec` value, the average value, and the average of the two previous values.

In comparing our algorithms, we consider both the average path length and the number of failed searches. (Searches can fail if an algorithm cannot backtrack and reaches a page with no outgoing links, or if there is no path between pages.)

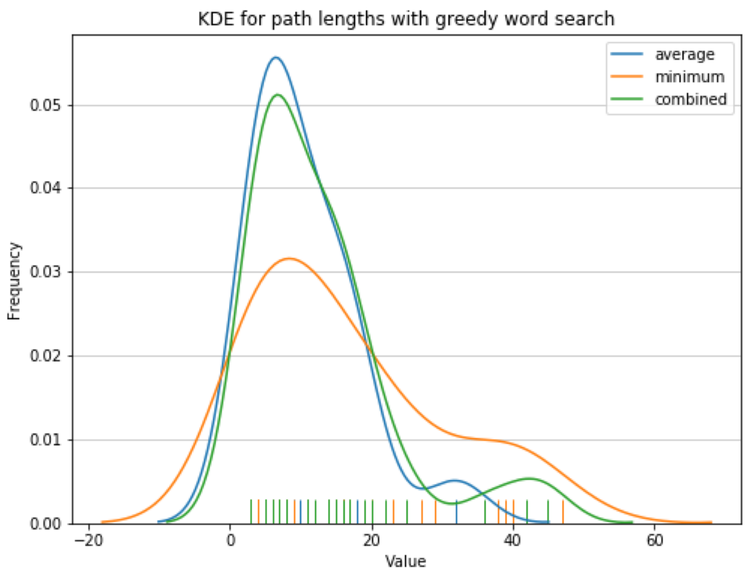


**Figure 1:** Kernel density estimate of average path length for greed word search

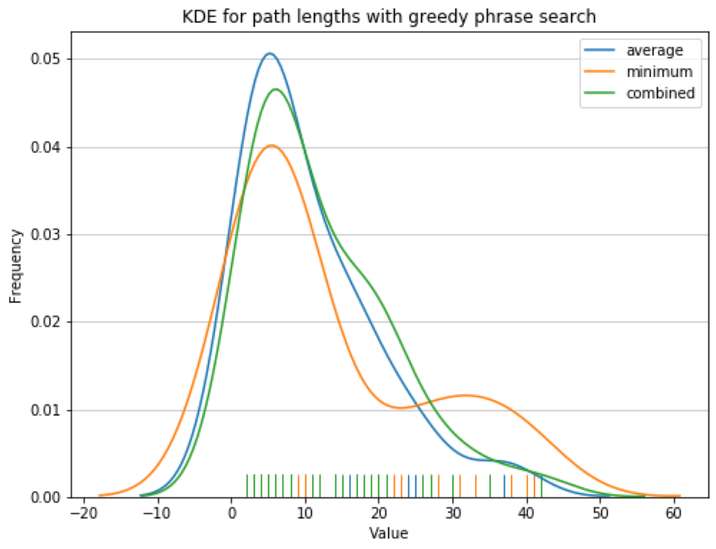


**Figure 2:** Kernel density estimate of average path length for greed phrase search

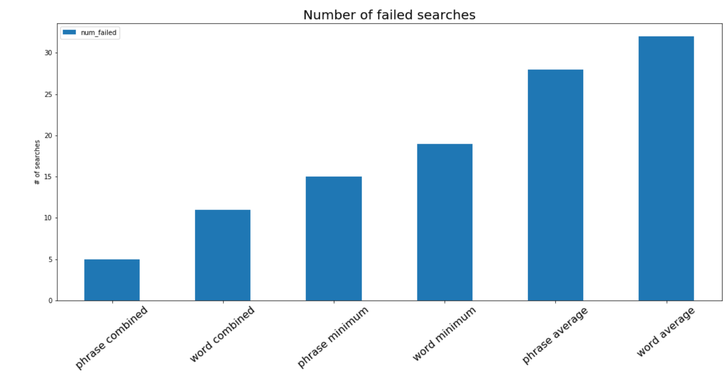


**Figure 3:** Bar graph of # of failed searches, by algorithm

## Conclusion

Our greedy heuristics performed similarly over successful searches with the average and combined approaches, while minimum word distance performed slightly worse. For minimizing failed searches, however, the combined approach performed by far the best. Incorporating phrases consistently improved results.

Moving forward, our first priority is to download and compile a subset of Wikipedia for faster search execution. The Wikipedia API is too slow to efficiently find the shortest paths using BFS when paths exceed 5 pages. BFS and bi-directional BFS will allow us to develop a path benchmark.

Next, we will focus on implementing A* and developing a admissible, and effective heuristic. Given that failed searches are more likely than previously believed, we will place greater emphasis on improving success rates moving forward.

## Sources

`code.google.com/p/word2vec/source/`

Jacob Wenger. Six degrees of wikipedia. `https://www.sixdegreesofwikipedia.com/`, 2018. Last accessed 12 November 2018.

Wikipedia: Six degrees of wikipedia. `https://en.wikipedia.org/wiki/Wikipedia:Six\_degrees\_of\_Wikipedia`, 2018. Last accessed 12 November 2018.