Creating a Path Finding Agent to Navigate Wikipedia

Chaitanya Asawa
Department of Computer Science
Stanford University
casawa@stanford.edu

Alexander Wang
Department of Computer Science
Stanford University
aswang96@stanford.edu

Jaimie Xie
Department of Computer Science
Stanford University
jaimiex@stanford.edu

Abstract

We aim to develop an agent that can navigate its way from a source node to a target node by traveling along edges, and with only knowledge about the current node and its outgoing edges. Specifically, we would like to develop this agent in the context of being able to navigate from one Wikipedia page to another, knowing only about the hyperlinks at its current page (a game known as Wikispeedia). To achieve this, we would like to explore using greedy decentralized search algorithms with some distance function, trying to measure the conceptual distance between nodes and additionally hoping to exploit graph properties that may lead to more favorable paths. We also attempt to understand how network characteristics affect our search ability. Finally, we try to see if we can use human intuition and strategies to boost our search performance as well.

1 Background

In this paper, we aim to create an agent that can find a path of links from a source Wikipedia article to a target Wikipedia article. We generally investigate pathfinding, and use Wikipedia and this game (known as "Wikispeedia") as context.

1.1 Relevant Work

West and Leskovec[2012] found that one common strategy used by humans in playing Wikispeedia was to navigate first to a hub page, and from there use similarity to the target to navigate from the hub to the target. Additionally, they found that the conceptual distance to the target decreased steadily at each node of the path. We made use of these insights about human intuition in path finding when implementing our own algorithm.

Kim and Hirtle[1995] define the Wikipedia graph as one that can be searched using "path" and "distance" strategies. "Path" strategy follows a procedural route description of how to reach the target. "Distance" strategy searches a fixed circle around a node's position, and can be used in conjunction with "direction" strategy, which uses a global frame of reference to travel towards the target destination. The greedy algorithm that we propose to use in our own algorithm is the "distance" and "direction" method referenced in the paper.

One problem with using the Wikipedia network directly is that the sheer number of hyperlinks obscures when two topics are actually closely related. West, Pineau, and Precup[2009] instead propose a method of measuring semantic distance using the "Wikispeedia" link paths that we have in our data set. We can exploit the possibility that these links provide more semantic significance in the analysis of our own semantic network, and use these distances for our greedy decentralized search.

1.2 Discussion

In West and Leskovec's paper, they perform a comprehensive analysis of efficiency and general characteristics of human path finding, although, while admittedly it is a very tough problem, it seems their ability to predict the target was not super successful. Improvements may potentially be made here by using different models or different features. In addition, there may be more to be learned about going from one link to the next rather than jumping to predicting the target node. Hence, our projects focuses on this area.

2 Problem

A huge problem that we face in the field of artificial intelligence is that of path finding, which is at the heart of the decision-making technology behind self-driving cars and AlphaGo. Our project considers path finding through decentralized search, and we hope some of the insights we gain in the context of Wikipedia can be generalized to more search applications. We also analyze the effectiveness of different ways of determining semantic distance and how graph structure plays a role in search.

2.1 Data Collection Process

The data we will use, released by the SNAP group at Stanford, consists of more than 30,000 instances of humans playing the Wikispeedia game. The Wikipedia graph used for these games covers about 4,000 articles and 120,000 links. In addition to these paths, we have the HTML and plain text versions of the Wikipedia articles. The dataset can be found at:

https://snap.stanford.edu/data/wikispeedia.html.

2.2 Algorithms and Models

The first algorithm we would like to develop and explore for navigation is greedy decentralized search with semantic distance. In a greedy algorithm of decentralized search, we would like to traverse edges that bring us to nodes that are closer to the target by some distance metric. We will explore and evaluate different semantic distance metrics such as tf-idf distance between Wikipedia pages, latent similarity, Jaccard Distance, and Sorensen Distance. We will then try to augment the capabilities of our search heuristics by using graph features such as out-degree, PageRank, and betweenness centrality. We will also incorporate strategies commonly used by humans into our algorithm, elevating it from a simple decentralized search algorithm. Using various combinations of distance metrics, graph feature heuristics, and additional human strategies, we will build and evaluate a variety of models.

2.3 Evaluation

We will evaluate the final performance of our various models by two metrics:

1) **Success Rate:** Every time an agent finds a path from the source to the destination under a predefined constant number of steps, we count this as a success. We will calculate, for each model, the number of successes over the number of attempts.

2) **Average Path Length:** For paths that are successful, we will compute, for each model, the average path length needed to find the target.

We will use these metrics for comparison across all models, and compare it to the actual shortest paths and the human intuition paths.

3 Methodology

We first ran our agent using four semantic different distance metrics to guide its pathfinding process. Our agent uses the greedy traversal strategy described previously, using the distance metric to choose the neighbor closest to the target destination at each decision point.

First, we preprocessed every Wikipedia page into a punctuation free, stemmed version of itself. Because we tokenize the pages into words based on spaces, punctuation characters create false unique words (i.e. "walk," will be seen as a unique word from "walk"). Stemming removes "er" and "ing" suffixes from words (i.e. "walking" will be stemmed to "walk"). Preprocessing allows our distance metric calculation algorithms focus more on the actual content rather than nuances in the English language.

Second, we made sure that the agent could not return to a node it has already visited, to avoid cycles. If any of the neighbors are the target node, for computational efficiency, we add it to our path and stop the search, rather than ranking the neighbor nodes (note though that our distance metrics would suggest the target node as the next node since the distance between the target node and itself would be 0). Additionally, we limited the number of steps that the agent could take to 10 (the search fails if the agent does not get to the target in 10 steps).

Finally, to actually select our source and target nodes, we considered 3 different selection methods. These selection methods are:

1) Selecting two random nodes. This helps us to understand our search performance in a very general context.

2) Selecting the start and end nodes from the start and end nodes of complete paths from humans who played the game. This helps us compare our performance to human performance.

3) Selecting the start and end nodes randomly such that they are articles that belong under the same Wikipedia category. This helps us understand our performance when traveling between a source and target that are guaranteed to have some similarity, in that they are listed under the same category.

For each of these 3 source/target selection methods, we ran 400 simulations for each of the distance metrics. We describe the semantic distance metrics in more detail in the section that follows.

After this, we added network features to our heuristic, weighting semantic distance and the network heuristic to create a new distance function that would guide our search. We first tried just the network heuristic by itself, and then combined it with the semantic distance, one heuristic at a time.

We also conducted analysis of different graph environments to see their influence on our search ability – looking at properties such as clustering coefficient, closeness centrality, and node eccentricity.

Finally, we used the human intuition of first going to a hub node and then trying to decrease conceptual distance to the target as a strategy in our algorithm.

4 Semantic Distance Metrics

4.1 Term Frequency-Inverse Document Frequency (tf-idf)

tf-idf has several variants based on specific weighting schemes. From a high level, tf-idf makes use of two pieces of information. First is term frequency. The more frequently a term is shared between two documents, the more similar they are. The second piece of information is inverse document frequency. The more rare a term is across an entire corpus of documents, the more significant it is when it is found to be common between two specific documents. The variant of tf-idf we use calculates idf (inverse document frequency) as follows:

$$\text{idf}(t) = 1 + \log \frac{1 + n_d}{1 + \text{df}(d, i)}$$

where n_d is the total number of documents, and df(d,t) is the number of documents that contain term t.

We create tf-idf vectors for two documents and compute the cosine distance between the two vectors – and this cosine distance is our final tf-idf distance.

4.2 Latent Semantic Analysis (LSA)

LSA first reduces documents to matrices, capturing each unique word's count per paragraph in the document. Singular value decomposition (SVD) is then used to reduce the number of columns within the matrices while preserving features, or meaning. Lastly, the cosine similarity between rows, or words, is calculated, giving a similarity metric for individual words, which can be aggregated to find the similarity between documents (and 1 - the similarity is the distance).

4.3 Jaccard Distance

Given two documents X and Y, we divide the number of unique, common words between the documents by the number of total unique words the documents contain. Formally:

$$1 - \frac{|X \cap Y|}{|X \cup Y|}$$

4.4 Sorensen Distance

Similar to Jaccard Distance; given two documents X and Y, we divide twice the number of unique, common words between them by the total number of unique words each document contains. Formally:

$$1 - \frac{2|X \cap Y|}{|X| + |Y|}$$

5 Comparison Across Semantic Distance Metrics

We present findings using our various distance metrics with different source and target node selection methods. First, we look at the success rates, which is the fraction of searches our agent successfully completed in the search limit of 10 steps.

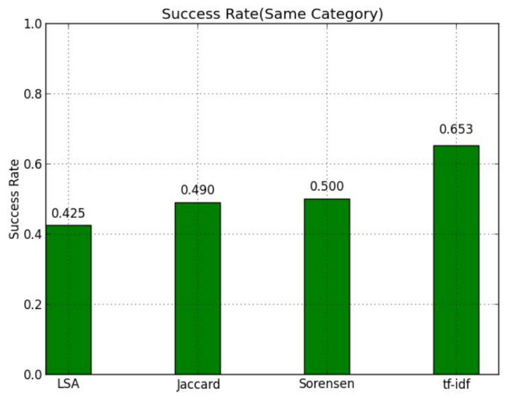


Figure 1: The source and destination are chosen randomly from the same category.

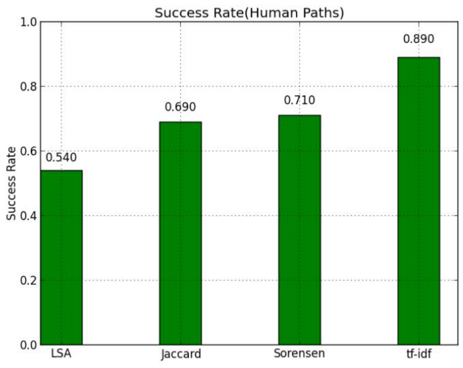


Figure 2: The source and destination are chosen from Wikispeedia instances.

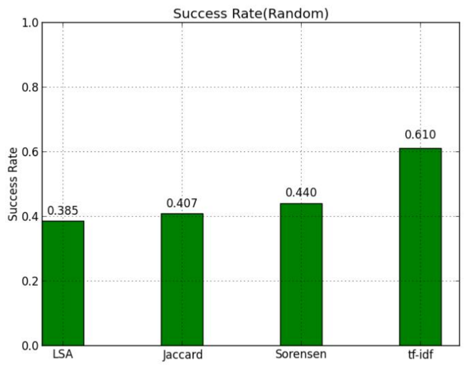


Figure 3: The source and destination are randomly chosen articles.

We see that tf-idf is far more successful in completing searches across all conditions. In particular, with tf-idf, we complete 89 percent of the searches when we use human paths for selecting our start and end nodes (note that humans complete 100 percent of these searches by definition). We also consistently see Sorensen performs the second best, followed by Jaccard, and finally LSA. Then, we looked at average path lengths for successful paths (when we say "True" we refer to the shortest possible paths):

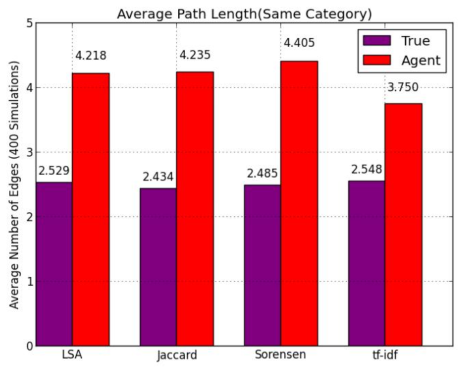


Figure 4: The source and destination are chosen randomly from the same category.

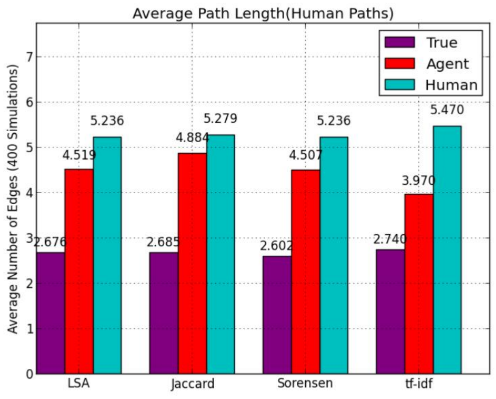


Figure 5: The source and destination are chosen from Wikispeedia instances.

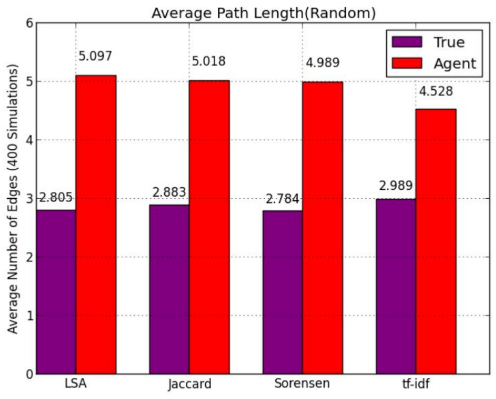
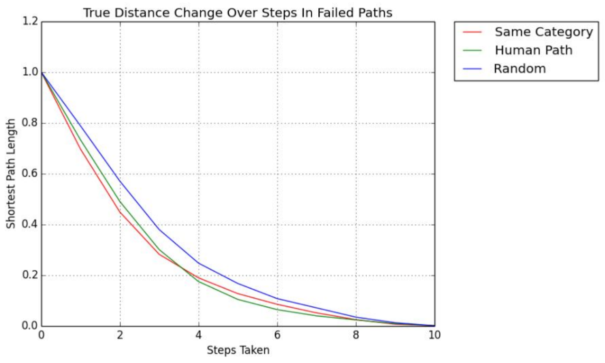
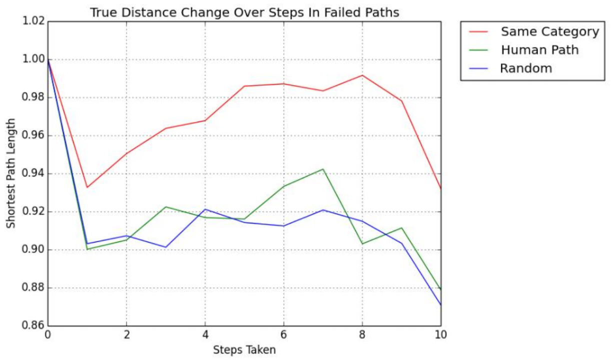


Figure 6: The source and destination are randomly chosen articles.

We see that tf-idf consistently across conditions takes, on average, fewer steps than the other distance metrics (about 0.5 steps less on average). Compared to the actual shortest path lengths, tf-idf seems to take 1.2 to 1.5 steps more. With Figure 5, we see that all of the distance metrics perform better than humans (at least, when our agent successfully finds a path; the humans were more successful in completion). tf-idf takes on average 1.5 less steps than humans.

Finally, we try to see, for successful paths and for failed paths, the average normalized, true distance to the target at a particular step. We compute this as follows: First, suppose the shortest distance between a source and target node is x steps. For normalization purposes, we consider the distance $x/x = 1$. Then, if after step 1, if the true distance is $x-1$ steps away from the target, our normalized distance is $\frac{x-1}{x}$, and so on. Using our most successful metric, tf-idf, we average the normalized, true distances to target for each step (if the path is that many steps long), separately for successful and failed paths, across all node selection methods.





We see that in successful paths, at each step, we are consistently getting closer to the target. However, with failed paths, with the first step we get slightly closer to the target, but for the most part, do not get any closer to the target with more steps (as our normalized distance at this point is always at least 0.9). This seems to suggest that failed searches make little to no progress on reaching the target, and we see the exact opposite for successful searches in that they consistently make progress. Interestingly, all failed paths seem to increase in distance after the first step, then begin to decrease distance at around 8 steps. This suggests that the first step in failed paths bring us to a local minimum, which our agent has to backtrack out of in order to correct. Some initial progress, then, seems highly likely to lead to a successful search. Going forward, we will then employ tf-idf as our measure of semantic distance due to its superior performance.

6 Using Network Heuristics in Search

tf-idf allows us to measure conceptual distance, but we may potentially augment our search capabilities by leveraging network features. For example, we may prefer to travel to a node of high degree, since it has more connections that could include the target or lead us to the target .

6.1 Single Network Heuristic

First, as a way to give insight on the usefulness of various network metrics, we use a single network metric to guide our search, similar to how we used only tf-idf distance previously. The metrics we consider are:

1) **Out Degree.** We may prefer to travel to nodes of high out degree since they have many connections – and hence it is more likely that one of these connections may be our target or lead to it.

2) **PageRank.** PageRank measures the importance of a page by assuming that more important website have more in-links. While this does not directly say anything about the out-links of a page and whether they will lead us to the target, we hypothesize that a Wikipedia page that has a high PageRank, so linked to by other important pages, most likely has a substantial amount of content (which is why it is referred to). Hence, we think it is likely to also have many connections. Additionally, a node with high PageRank may be a broad topic, which is useful potentially to help prevent going down a narrow topic path. The broadness and connections of high PageRank nodes may improve our search performance.

In our Wikipedia graph, we determined the nodes with the highest PageRank are "United States", "France", "Europe", and "United Kingdom". Indeed, this seems to validate our hypothesis that these pages have many out-links and are broad concepts. In particular, they seem to all be countries, and humans actually tend to incorporate Geography often in their searches even when geography is not their target [1]; hence, PageRank may help capture some form of human intuition of broadness of a concept.

3) **Betweenness Centrality.** Betweenness centrality of a node is the number of times a node is in the shortest path between two other nodes, and hence we would prefer these nodes since we are looking to travel along the shortest path between the source and target.

In our searches then, we will always try to go to nodes with the highest out degree, PageRank, or betweenness centrality.

6.2 Single Network Heuristic Results

Using the previously described network features to solely guide our agent, we ran a series of experiments on the resultant success rates and average path lengths.

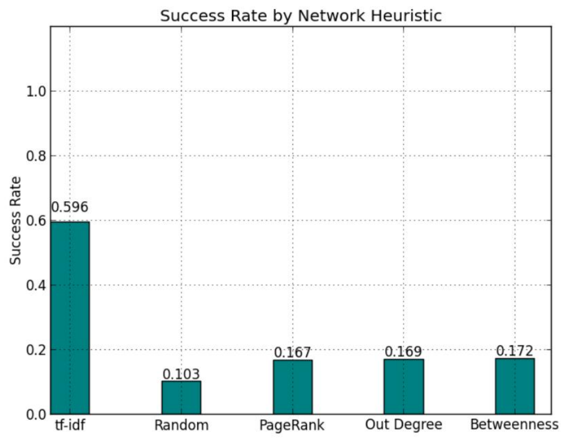


Figure 7: The success rates of our agent, using different network heuristics.

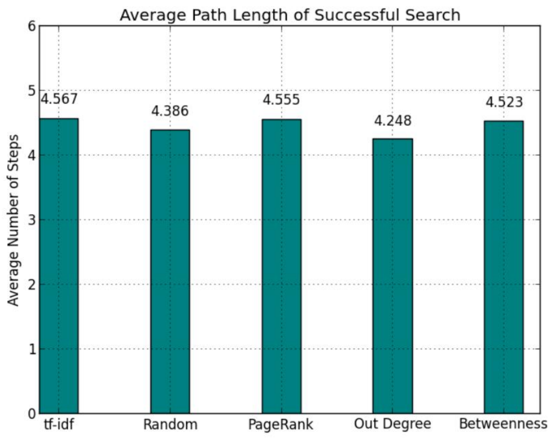


Figure 8: The average path lengths of successful searches with different network heuristics.

As expected, none of these metrics by themselves perform at the level of tf-idf, since they do not even consider any information about the target to guide their search. However, it seems that these metrics do perform better than random – specifically, they all lead to around the same success rate of 0.17, whereas a random model that chooses random edges (until the target is a neighbor of the current node the agent is on) has a success rate of 0.1. The average path lengths of successful searches seem to be roughly the same – perhaps indicating that the various heuristics do have the ability to choose short paths, but have lower success rates because more often than not they go in a wrong direction.

6.3 Combining Features

We then explored combining tf-idf distance with graph properties – trying to leverage tf-idf to help us measure conceptual distance and actually incorporate information about the target, while exploiting graph properties to choose nodes that are more likely to be connected or on shortest paths to increase search efficiency. Hence, we hypothesize that some combination of tf-idf distance and graph properties will allow our agent to perform better than using tf-idf on its own.

Since our tf-idf and network features values are at very different scales, we normalized our features to have zero mean and unit variance as follows:

$$x' = \frac{x - \mu}{\sigma}$$

where x is the original value of the feature, μ is the mean value of the feature, σ is the standard deviation of the feature, and x' is the normalized value.

We then combined tf-idf with each network heuristic one at a time, exploring different weighting schemes. In particular, we defined the distance to the target to be

$(1-w)*$(normalized tf-idf)$+w*$(normalized network heuristic)

where w is a weight between 0 and 1.

Note that for certain network heuristics, such as out degree, we would like to travel to nodes that have a high value for out degree. As our search tries to minimize distance though, we multiplied out degree by negative 1 such that minimizing negative out degree would lead to a node of high out degree.

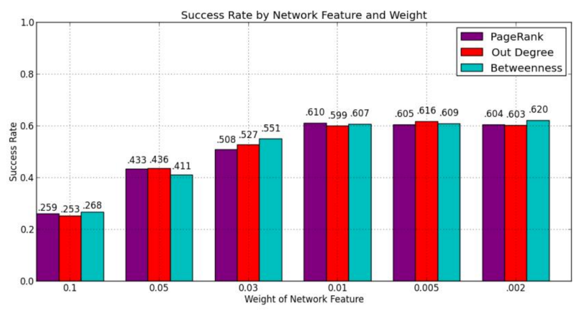


Figure 9: The success rates of our agent, using different network feature with varying weights.

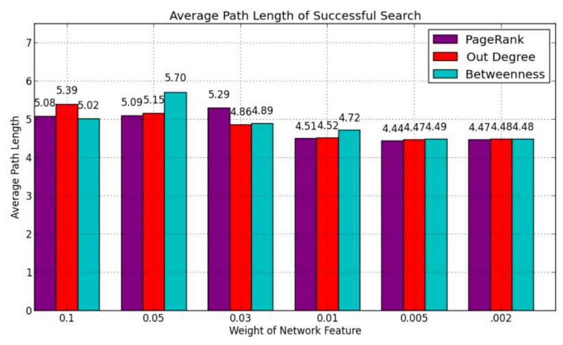


Figure 10: The average path lengths of successful searches using different network features with varying weights.

The success rate of our agent reaches as high as 0.62, which is marginally better than the pure tf-idf agent's performance of 0.60. We theorize that tf-idf is by far the most useful metric to guide our agent with, and additional network features can act to break ties between nodes of similar tf-idf distances. For example, if the node has two neighbors with similar tf-idf distances from the target node, it should choose the neighbor that has a greater out-degree, as it is more likely to lead to a successful path.

The reason for only marginal gains may be that Wikipedia is already considered a "small world" graph with most pairs of nodes connected by short chains [1], and hence in our problem leveraging graph structural information for finding shorter and successful paths is not as useful as using conceptual distance to arrive at our destination.

We plot the out-degree distribution for our Wikipedia graph and we indeed see a similar shape to the degree distribution for a small world graph:

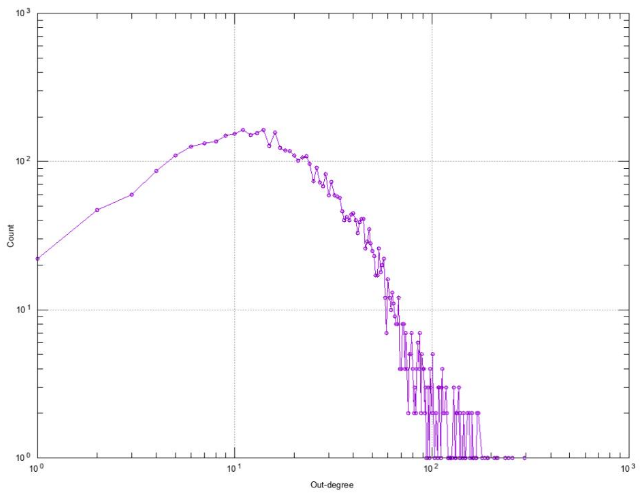


Figure 11: Out Degree Distribution of Wikipedia Graph.

7 Comparison Across Search Environments

We are interested in seeing how well our agent performs in different environments – specifically how do different graphs affect our ability to conduct successful searches. To create these varying environments, we subgraph the Wikipedia graph by different article categories, creating a subgraph for each category. These category graphs have differing characteristics that influence our search ability. We compute the number of nodes in each subgraph, as this has a large influence over the graph properties:

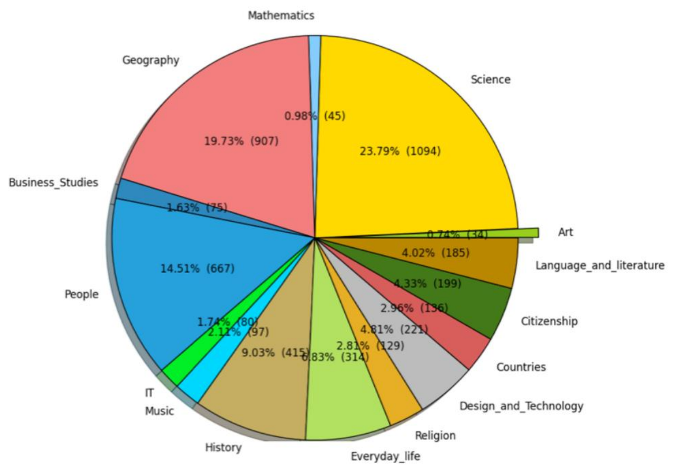


Figure 12: Percentage and number of graph nodes by category.

We then run our agent over these different subgraphs, computing success rate and average path length. For the average path length of the subgraphs, we divide by the effective diameter (an approximation of the graph's diameter) to normalize these path lengths across different graphs.

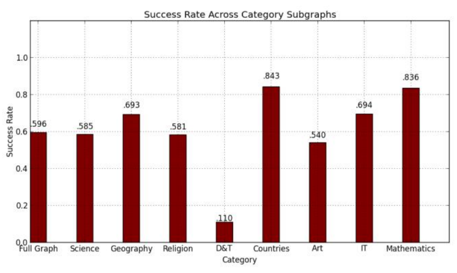


Figure 13: The success rates of our agent across different category subgraphs.

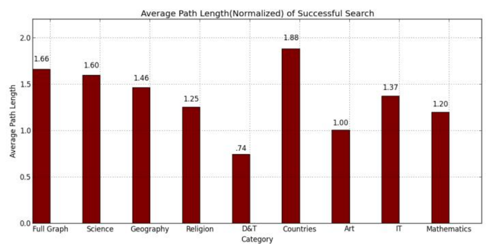


Figure 14: The normalized average path lengths of successful searches, normalized by dividing out the subgraph diameter

It seems that success rates and average path lengths when successful are correlated – that is, for graphs where the success rate is low, when the agent does succeed, it does not use many steps. For graphs with higher success rates, the agent uses more steps when it succeeds. This seems to indicate that in "less navigable graphs" (those with lower success rates), it is hard to be successful unless the target is close, but in more navigable graphs we eventually manage to find our way. We will try to then more concretely determine which graph properties influence this.

In our analysis, we see that Mathematics and Countries have high success rates, and Design and Technology (D&T) has a low success rate, and so we will try to focus our investigation using these graphs. We look at average clustering coefficient of the nodes, average closeness centrality of the nodes, and the average node eccentricity of the nodes.

7.1 Average Clustering Coefficient

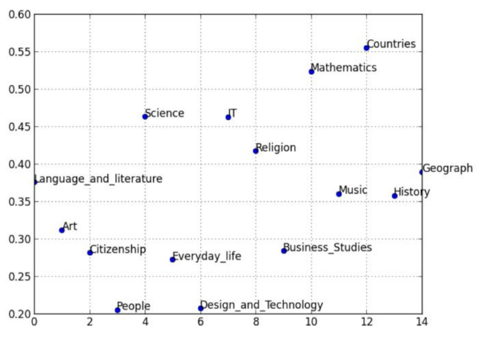


Figure 15: Average clustering coefficient across different category subgraphs.

We see that Mathematics and Countries have a high average clustering coefficient, and Design & Technology has a low average clustering coefficient. This makes sense, as we expect graphs that are more clustered are more navigable since there are many connections that bring the nodes in the graph together.

7.2 Average Closeness Centrality

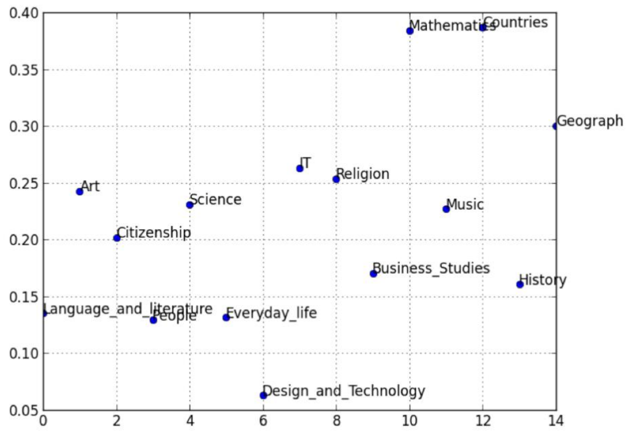


Figure 16: Average closeness centrality across different category subgraphs.

With average closeness centrality, we see that Mathematics and Countries have a high average closeness centrality, and Design & Technology has a low average closeness centrality. This is according to our expectations, as graphs with a higher average closeness centrality have shorter distances between their nodes, increasing the likelihood for an agent to traverse the distance between nodes.

7.3 Average Node Eccentricity

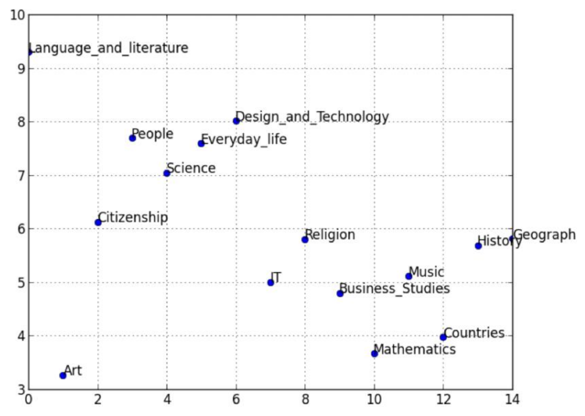


Figure 17: Average node eccentricity across different category subgraphs.

We then juxtapose average node eccentricity across the graphs, and note that Mathematics and Countries have a low average node eccentricity, and Design & Technology has a high average node eccentricity. The eccentricity of a node is a measure of the maximum distance between it and any other node. The above graph makes sense, as a low average eccentricity indicates, similar to high average closeness centrality, that nodes are generally close together, without many outliers that have large distances between them.

7.4 Mathematics Subgraph

We visualize the mathematics graph, displaying how clustered it is, and this seems to contribute to its navigability:

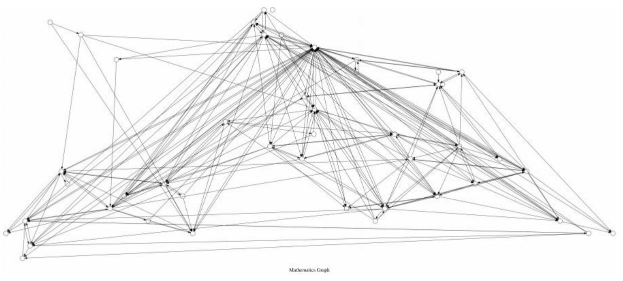


Figure 18: The Mathematics Subgraph

By comparing graphs of different properties, we see that factors such high clustering, high closeness, and low node eccentricity all seem to be correlated both with each other, and also correlated with higher success rates.

8 Search Strategies

A common human strategy when playing Wikispeedia is locating a hub (finding a node of high degree in their first click), and then from there continuously decreasing their conceptual distance to the target after [1].

We tested this strategy, which we have dubbed the "hub then hone" strategy, with our agent – jumping to the highest degree neighbor in the first edge traversal, and from there using the shortest tf-idf distance to the target to guide the rest of the search. We show the success rates and the normalized average path lengths across different categories, with and without this strategy:

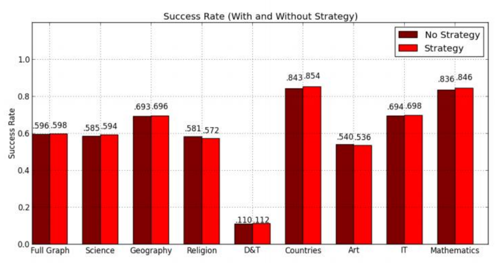


Figure 19: The success rates of our agent across categories, with and without the "hub then hone" strategy.

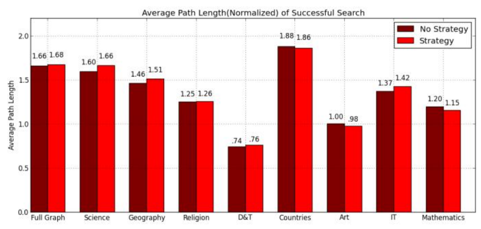


Figure 20: The average path lengths of successful searches with and without the "hub then hone" strategy across categories, normalized by dividing the subgraph diameter

It seems that using the strategy did not make a difference, across the various categories, for success rate or average path length. Since the average path length does not change, it seems that the value of traversing to a node with high degree on the first step is the same value as conceptually getting closer. To measure this utility, we graph the normalized true distance change over steps, as we did in section 4.

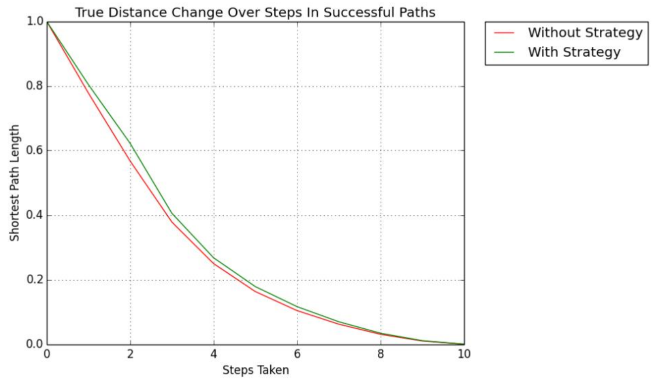


Figure 21: The average distance of our agent to its target page over a successful path, with and without the "hub then hone" strategy

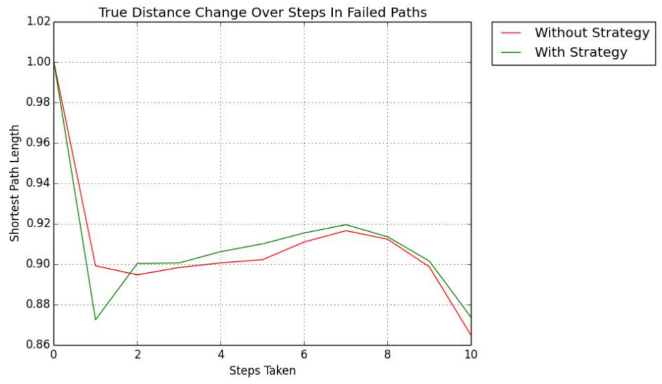


Figure 22: The average distance of our agent to its target page over a failed path, with and without the "hub then hone" strategy

We find that the normalized true distance over steps curves for successful paths with and without the strategy are nearly identical. It seems that a node of high degree (since it has many connections that could lead to short paths to our target) brings us as close to the target as the first step with using conceptual distance. Hence, while this strategy may benefit humans, as it might be easier to determine broad concepts and hubs rather than conceptual distances for all out-links, our algorithm does not benefit in successful cases since it has the ability to compute for every single out-link a heuristic for conceptual distance.

For failed paths however, we see that going to a hub first brings us closer to the target on the first step than solely using tf-idf. This makes sense as we saw previously that the first step is important to the success of a search, and that tf-idf on the first step seems to go down the wrong path so its normalized true distance will be high, whereas going to the highest degree node is more concept independent and has many connections to what could be viable paths so its normalized true distance will be lower. Therefore, it may actually be better to use the strategy, since the first node gives us greater flexibility of paths that we can take since it has a high number of out-links and does not negatively impact our true distance to the target or our success rate.

9 Challenges

Computing the semantic difference between two articles is still an open problem, and our project heavily relies on such methods when selecting the next node to travel to. Furthermore, when evaluating how “close" our agent is to the target, the best metric of “true" distance we have is the shortest path between nodes. Because of how highly connected the Wikipedia graph is, this is a very rough evaluation metric (this issue was discussed by West, Pineau, and Precup).

In addition, the graphs we had were relatively small, and perhaps using larger graphs could help better develop the use of network heuristics, as we saw only marginal gain in our already fairly well connected graphs.

10 Conclusion

We have found that tf-idf leads to more successful searches and, on average, takes fewer steps than all other semantic distance metrics. Additionally, all the distances metrics, in successful searches, seem to take fewer steps than humans. We also see that in successful searches the agent seems to consistently make progress, whereas in failed searches the agent makes very little to no progress. This seems to indicate that initial progress may be a good indication of whether a search will be successful.

We then hypothesized that leveraging network characteristics along with tf-idf would substantially improve our search performance, but we concluded this hypothesis to be mostly false. While these network characteristics are more useful than taking random edges, it seems that tf-idf is far more useful in search in this case, potentially since Wikipedia is already a “small world" graph.

We also found that different graph structures can substantially affect the search success. We see that factors such high clustering, high closeness, and low node eccentricity all seem to be correlated both with each other, and are also correlated with better success rate.

Finally, we also investigated using a common human search strategy of first going to a hub and then honing in on the target conceptually. We found our agent performs the same in terms of success rate and average path length with and without using this strategy across a series of categories. However, it may still be beneficial to use this strategy, as it may afford more flexibility for paths initially rather than going down potentially a narrow wrong path.

11 Next Steps

Going forward, it may be interesting to see if our results generalize to other portions of the Wikipedia graph or larger portions. In addition, it may be interesting to see how our agent performs in other networks of articles. For example, the world wide web, graphs of news articles that link to each other (as news websites very often refer to previously news articles in current news articles), or perhaps even the LinkedIn profile network can be thought of as a network of articles.

We were not able to successfully leverage graph properties much, but more exploration can be done here, perhaps with combining more graph features. Machine learning methods can potentially be used to learn the weights of these more complicated sets of features. These methods may first be used just to rank a individual node without using history, and then perhaps using sequential models such as recurrent neural nets, LSTMs, and GRUs with the previous nodes in a path to more effectively determine the next node. We would hope that these methods can capture some complex, nonintuitive rules that can be used to boost our search efficiency.

12 Contributions

All group members contributed equally.

13 References

[1] West, Robert, and Jure Leskovec. “Human path finding in information networks." Proceedings of the 21st international conference on World Wide Web. ACM, 2012.

[2] Kim, Hanhwe, and Stephen C. Hirtle. “Spatial metaphors and disorientation in hypertext browsing." Behaviour & information technology 14.4 (1995): 239-250.

[3] West, Robert, Joelle Pineau, and Doina Precup. “Wikispeedia: An Online Game for Inferring Semantic Distances between Concepts." IJCAI. 2009.