# wikiwebwalker: Using Semantic Similarity Analysis to Optimize Finding Paths between Articles in an Interlinked Database

**Jasmine Cai, Nate Courchesne** and **James Grassi**
Manning College of Information and Computer Sciences
University of Massachusetts Amherst
Amherst, MA 01003
{jlcai, ncourchesne, jgrassi}@umass.edu

## Abstract

To optimize searching for a path between specific topics in a vast interlinked database like Wikipedia, we need to use various search techniques that will help speed up the process. We used a unique natural language processing metric, semantic similarity, as a measure for determining the relationships between words in articles in a database. In this paper, we proposed a streamlined way of using this measures to aid in Wikipedia traversal by it's use as heuristics to aid in an assisted graph search algorithm. We tested several search algorithms like breadth-first search and A* against Wikipedia articles and evaluated each pass based on how few steps it takes to reach point B from point A.

## 1 Introduction

### 1.1 Problem Statement

The Internet continues to grow every day and, as a result, looking for specific articles or topics in such a large database of knowledge becomes very difficult without the aid of tools. With the vastness of it all, there are many instances where pages never link together, and it becomes easier to lose various pieces of information due to a lack of indexing and accessibility. Thankfully, within streamlined and interlinked databases like Wikipedia, this problem becomes less of an issue. The main problem now lies in finding paths from article to article, where we need to find the links between different topics on different sides of Wikipedia. Finding the missing links between different articles can be very important for research to help bridge connections to prove certain path solutions feasible due to their unapparent relatedness.

### 1.2 Background

Large databases of knowledge like Google utilize various search techniques and query "expectation" metrics to help provide relevant queries for their users every day.

Common search and data extraction techniques involve web crawling, API integration, optical character recognition (OCR), and other forms of database querying (Docsumo, 2024). Google utilizes web crawling where bots crawl pages, and each page goes through a preprocessing phase where all of the textual content, attributes, page elements (i.e. HTML), and more are parsed and indexed (Google, 2024).

Natural language processing metrics like semantic similarity and contextual relatedness are frequently utilized in search optimization and are good ways to find relationships between different words. Semantic similarity is a measure of how frequently words co-occur with each other (i.e. synonyms or near-synonyms). Contextual relatedness is a measure of how frequently words occur within a similar context (related by subjects they discuss). These metrics can be used on word embeddings of word documents to determine their approximate similarity. Methods like **cosine similarity** are a common way to see the semantic similarity of two different word documents $\mathbf{X}$ and $\mathbf{Y}$ and measure the angle between the two vectors (Wikipedia, 2024). The equation for cosine similarity can be seen here:

$$\text{cosine similarity} = \frac{\mathbf{X} \cdot \mathbf{Y}}{||\mathbf{X}||||\mathbf{Y}||} \tag{1}$$

### 1.3 Proposal

This report documents our approach to solving this interconnecting problem by utilizing natural language processing metrics like semantic similarity to aid graph searches between different articles, or nodes, in a database web. We used this metric to guide our graph search to minimize the number of steps it takes from point A to point B in this interlinked network. Our approach consists of two main components: devising the network graph search algorithm and determining semantic similarity embeddings.

For devising the graph search algorithm, we wanted to use algorithms that can step into links within an entry-point article, gather information about each link within it, and ideally make an educated step closer to the exit-point article based on our aforementioned NLP metric. After determining which types of algorithms would fit the most with this behavior, we compared **breadth-first search**, our control group, to an **A* search** using semantic similarity between articles as its heuristic function. Both algorithms prioritize finding the shortest path given the distances (or hops) between nodes. From any given Wikipedia article, it is possible to view the title and title of all articles that they link to. After obtaining all the links within a layer, we have three methods of utilizing the articles in our semantic scoring algorithm: full documents (articles), the first paragraph of articles, or article titles only.

For the semantic similarity measure, we planned to utilize large language models skilled in taking the word embeddings of documents like **DistilBERT** to implement similarity algorithms like **cosine similarity**. After obtaining the embeddings of articles using pre-trained DistilBERT models, we can use them and evaluate their similarity scores against the exit point article via cosine similarity.

After testing these two search algorithms and three different methods of using the Wikipedia corpus, we aim to learn if we can efficiently determine the least number of steps between two different points in an interlinked database, and if measures like semantic similarity are good metrics of weighting in our search.

## 2 Related Work

### 2.1 Enhanced search via semantic similarity queries

Topic-specific web crawlers usually focus on link analysis (determining the most popular links via backlinks, content analysis (relevant web pages to what your main goal is), and ontologies (removing irrelevant pages and subsequently improving performance). (Pesaranghader et al., 2013) suggests a new method of training web crawlers that improves on something previous crawlers do not touch upon: considering semantic similarity alongside the "sense", or the definition regarding the original context, of the input topic word.

They present the idea of using the visualization of documents as vectors in the space of terms via Vector Space Models (VSMs) in Information Retrieval (IR) for determining semantic similarity. To measure this similarity, they utilized cosine similarity.

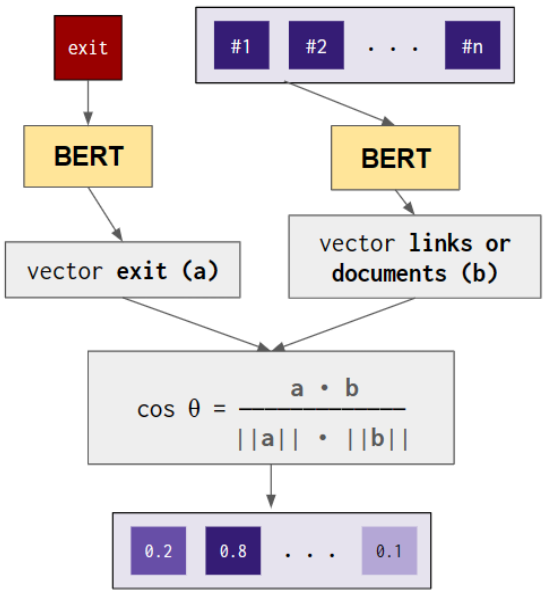


Figure 1: Determining semantic similarity (via cosine similarity) of a vector space of $n$ documents compared to a webcrawler query document (exit).

Our first idea when tackling the problem of interlinked database traversal relied mostly on web crawlers and Selenium webdrivers. We later opted for utilizing the static HTML files instead, but we still used this paper as a good example of considering semantic similarity (cosine similarity as a numerical metric) for enhanced search.

### 2.2 Efficient crawling via URLs with ordered frontlines and backlinks

Creating an ordered queue of links to go through with a webcrawler or database network search would be beneficial for efficiency and computational complexity. With a problem like traversing Wikipedia, the computational complexity of a search through it all will only exponentially increase with the increased vastness of the database subset we work with. (Cho et al., 1998) describes different efficient URL sorting algorithms with enqueuing and dequeuing frontlinks and backlinks to find some "hot" page. This paper describes general metrics for how to develop efficient web crawlers and tips on how to create an ordering strategy. This paper may become useful because the computational complexity of what we're trying to accomplish will only exponentially increase with the vastness of the subset of Wikipedia we are dealing with,

so having some ordering strategy and methods of increased efficiency will prove to be useful.

The crawling and semantic search algorithms that are described here all stop after visiting $K$ URLs. We seek to adapt these types of algorithms until we reach a known exit-point article.

### 2.3 Breadth-first search of Wikipedia

(Wenger, 2018) is a website that has a very similar task to ours. Their main search algorithm is breadth-first search, but it does not seem like there is a heuristic based on any natural language processing-based heuristics. This inspired us to attempt to do breadth-first search and see if it was feasible to step through the links of Wikipedia only with frontlinks and backlinks, since this tool appears to demonstrate a functional implementation of the concept.

## 3 Data

Our dataset is composed of the pre-processed and extracted Plaintext versions of Wikimedia Downloads (a.k.a WikiDumps). This database was obtained from Wikipedia's database backup dumps, and includes a total copy of all Wikimedia wikis, i.e. the wikitext source and XML-embedded metadata of each page. We decided to choose the XML files instead of static HTML because the file size of our dataset is 23 GB uncompressed and it would take a very long time to download the HTML version. The zipped XML file is already 20.5GB, and XML allows the transfer of data to go smoothly. In addition, since XML is very similar to HTML in the sense that they are both markup languages, the XML file can be parsed easily to look for our desired information.

To preprocess and extract the WikiDumps XML file we made use of Wikiextractor, which is a command-line tool that extracts data from Wikipedia dump files. It can run as a subprocess and directly extract files into a directory. Afterward, the directory can be accessed and parsed to convert the files to a list of strings. Next, the JSON data can be loaded from the files to get a list of dictionaries, which can then be used for preprocessing. While obtaining all the articles could be done with many GET requests via Python libraries like **Wikipedia** (aptly named), we did not want to overload the Wikipedia servers with mass amounts of scraping. Utilizing the static files themselves allowed us to avoid any unnecessary errors with sending requests to obtain the data i.e. bandwidth consumption caused by trying to request such a large amount of data.

Each pre-processed and extracted file given by Wikiextractor contains many articles, so we parsed each file using regular expressions. With each article, the HTML was preserved and internal links could easily be extracted. In total, we have 237 folders each with 100 files, and each file with anywhere from 200 to 1,000 articles. This totals about 22.8 GB of data and 16,000,000 articles – all of which were stored on an external SSD. We used a smaller subset of files ( 2,200 files) for training demo purposes which will be detailed in the Results section.

## 4 Method

From any given Wikipedia article, it is possible to view the title and short descriptions of all articles that it links to. We implemented both breadth-first search and A* search. We also measured the difference between three different metrics: article titles, article's first paragraph, or article's full document text to determine which one is faster and takes fewer hops. Once the testing metric was chosen, upon visiting an article, all of the articles it links to are ranked by their likely similarity to the destination article we are trying to reach, by calculating the semantic similarity of the metric in question to that of the destination article. These algorithms will then open links from their previously traversed articles starting with the most-semantically-similar first. These algorithms were assessed to determine how much more efficient they are, if at all, than other graph traversal-based approaches, like Breadth-First-Search, or searches that prioritize other metrics such as checking larger articles first.

**Algorithm 1:** Semantic Similarity

```
1 Function most_similar(entry, current_layer)
    is
2     similarity_array = [] ;
3     foreach node: current_layer do
4         similarity = cosine_similarity(entry,
            node) ;
5         similarity_array ← similarity
6     return max(similarity_array) ;
```

**Algorithm 2:** A* Graph Search with Semantic Similarity Heuristics

```
Function A_star(entryName, exitName) is
    visited = ()
    queue ← page names, scores
    while True do
        queue ← sorted(scoresinqueue)
        path ← most recent in queue
        currNode ← most recent in path
        currPage
          ← currNode's pageName info
        if currPage is exitName then
        return path
        if currPage not in visited then
        is exit page visited ← currPage
        if links not empty then
        most_similarity of article
        names/titles,
        article first paragraph,
        OR article full doc
        visited ← most sim page
    return path
```

## 5 Results

For calculating results, we utilized the *Six Degree of Wikipedia*'s number of links traversed (hops) and the time elapsed as a benchmark to compare our implementation of the task to. We included the table and graph representation of the data to help visualize the aggregated graph (Fig. 7) easier since it is printed very small.

Ultimately, breadth-first search was too slow and did not produce any meaningful data (or any data at all), so we measured based on our A* approaches with different amounts of the corpus considered for semantic similarity.

We ran five different Wikipedia traversal tasks: Judge Dredd to North Korea, North Korea to Japan, Judge Dredd to Samurai, Japan to Shogun, and Japan to Samurai. For each of these traversal tasks, we used the article titles, first paragraphs, and full documents of the interconnected nodes as the basis of our cosine similarity.

Overall, the results seem to indicate that utilizing the titles of articles only when doing our Wikipedia traversal seems to be the fastest, while the first paragraph and full document tend to take longer than each other in some experiments without a clear indicator as to why. For the number of hops, it seems like all of the different experiments that we performed (disregarding the baseline SDOW) had a comparable amount of hops, each differing with $\pm 1$ hop across each traversal trial.

| SDOW | Titles | 1st Para | Full Doc |
|---|---|---|---|
| 3 hops | 5 hops | 8 hops | 6 hops |
| 1.03s | 5.65s | 240.8s | 135.65s |

Table 1: Number of links traversed (hops) and time elapsed for Six Degrees of Wikipedia (SDOW), A* search w/ semantic similarity of only article titles, first paragraph, and full document respectively of *Judge Dredd → North Korea.*

| SDOW | Titles | 1st Para | Full Doc |
|---|---|---|---|
| 1 hop | 2 hops | 2 hops | 2 hops |
| 0.49s | 0.7s | 50.2s | 58.6s |

Table 2: Number of links traversed (hops) and time elapsed for Six Degrees of Wikipedia (SDOW), A* search w/ semantic similarity of only article titles, first paragraph, and full document respectively of *North Korea → Japan.*

| SDOW | Titles | 1st Para | Full Doc |
|---|---|---|---|
| 2 hops | 7 hops | 6 hops | 6 hops |
| 1.07s | 4.6s | 187.5s | 183.2s |

Table 3: Number of links traversed (hops) and time elapsed for Six Degrees of Wikipedia (SDOW), A* search w/ semantic similarity of only article titles, first paragraph, and full document respectively of *Judge Dredd → Samurai.*

| SDOW | Titles | 1st Para | Full Doc |
|---|---|---|---|
| 1 hops | 2 hops | 2 hops | 3 hops |
| 0.44s | 0.8s | 51.1s | 92.1s |

Table 4: Number of links traversed (hops) and time elapsed for Six Degrees of Wikipedia (SDOW), A* search w/ semantic similarity of only article titles, first paragraph, and full document respectively of *Japan → Shogun.*

| SDOW | Titles | 1st Para | Full Doc |
|---|---|---|---|
| 1 hop | 3 hops | 3 hops | 3 hops |
| 0.42s | 2.4s | 125.7s | 83.4s |

Table 5: Number of links traversed (hops) and time elapsed for Six Degrees of Wikipedia (SDOW), A* search w/ semantic similarity of only article titles, first paragraph, and full document respectively of *Japan → Samurai.*

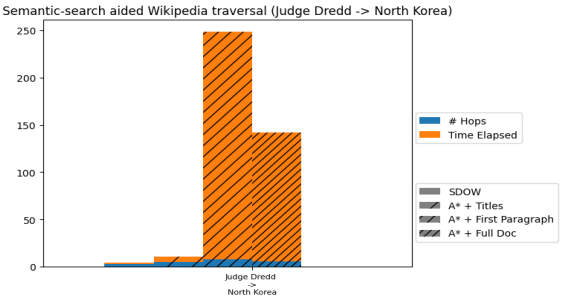


Figure 2: Graph comparing number of hops and time elapsed between SDOW (baseline) and our three experiments of A* search with article titles, article first paragraphs, and article full documents for *JudgeDredd → NorthKorea.*.

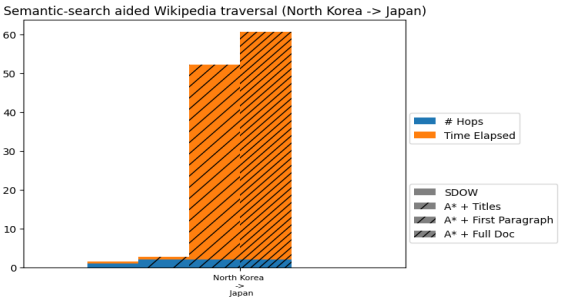


Figure 3: Graph comparing number of hops and time elapsed between SDOW (baseline) and our three experiments of A* search with article titles, article first paragraphs, and article full documents for *NorthKorea → Japan.*.

## 6 Discussion and Future Work

While our work demonstrated that using semantic similarity as a heuristic for an A* search could lead to speed and computation improvements compared to Breadth-First-Search, we ultimately failed to improve on the existing speed and accuracy of Six Degrees of Wikipedia due to other factors including less optimized code and a lack of caching. Re-implementing our search algorithm with a different code base which utilizes other search-time optimization techniques would allow us to effectively demonstrate the practical applications of semantic A* search.

We also did not ultimately achieve our initial goal of comparing algorithms based on both semantic similarity and semantic relatedness. As other studies on semantic relatedness such as (Abdalla et al., 2023) have noted, there is a current shortage of data and resources on the topic of semantic relatedness. In particular, the lack of an available model that could be slotted into our code which was able to calculate the semantic related-

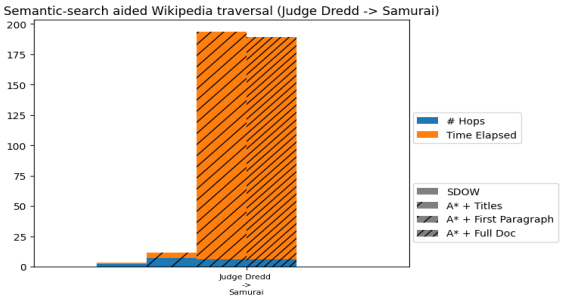


Figure 4: Graph comparing number of hops and time elapsed between SDOW (baseline) and our three experiments of A* search with article titles, article first paragraphs, and article full documents for *JudgeDredd → Samurai.*.

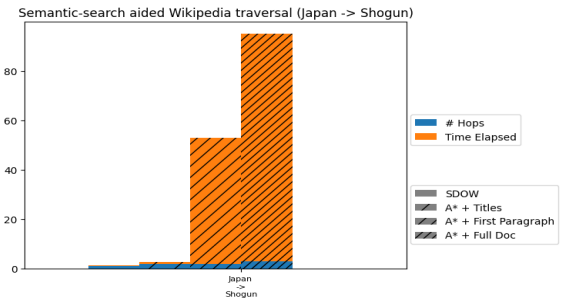


Figure 5: Graph comparing number of hops and time elapsed between SDOW (baseline) and our three experiments of A* search with article titles, article first paragraphs, and article full documents for *Japan → Shogun.*.

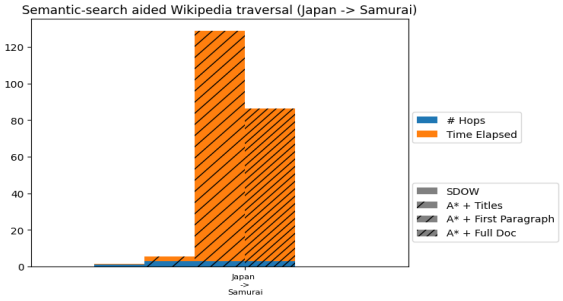


Figure 6: Graph comparing number of hops and time elapsed between SDOW (baseline) and our three experiments of A* search with article titles, article first paragraphs, and article full documents for *Japan → Samurai.*.

ness of two words in the same way that we could for semantic similarity put the concept outside the immediate scope of our project. If implemented, a search based on semantic relatedness could potentially further reduce the number of articles that our search algorithm needs to check before finding its target article. The shortest paths between articles often involve traversing through articles that are

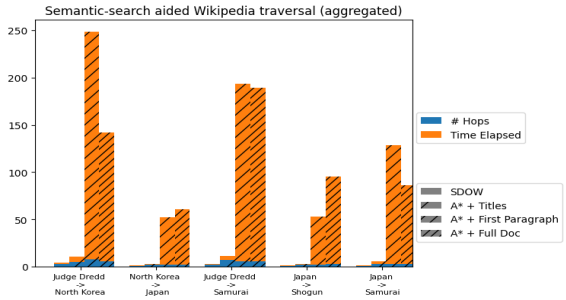


Figure 7: Aggregated graphic comparing the number of hops and time elapsed between SDOW (baseline) and our three experiments of A* search with article titles, article first paragraphs, and article full documents.

about different subjects within the same field, but do not necessarily mention the subject of the target article or any close analogues thereof. Some of the benefit of this style of search, however, is already captured by the version of semantic A* which takes into account the entire text of the article, since any two articles about subjects in a similar field are liable to share some of the terminology related to that field.

In addition, we found that two issues with our dataset were likely to have caused problems with out results. First, we could not use the entirety of the WikiDumps dataset due to the size. Our parsing algorithm could not go through the files in a reasonable amount of time, so only a subset was used. As a result, many links might not have any data associated with them, so they can not be considered during traversal. Therefore, the most efficient path from one page to another may not be currently exist. With the addition of every link, our algorithms may be a lot quicker and use less hops. Second, the XML files provided by WikiDumps often had gaps in data. When considering Six Degrees of Wikipedia, "North Korea" to "Japan", "Japan" to "Shogun", and "Japan" to "Samurai" can be found in only one hop. This is because each of the starting pages have the exit page as a link. Since our algorithms should be using the same dataset, we should expect one hop to occur as well. However, as the results show, this is not the case. For example, when looking in the "Japan" XML file, many paragraphs do not exist when compared to the actual Wikipedia page. In particular, both "Samurai" and "Shogun" do not exist in the XML file, while they do in the Wikipedia page. This causes our algorithms to make an extra hop before finding the exit page from another source page. In the future, if we were able to use a more reliable dataset, it is expectant for our algorithms to run in much more efficient manner.

Despite these shortcomings, it is clear that searching based on semantic similarity can reduce the number of nodes that need to be checked when traversing between two articles in an interlinked database of texts which connects related items.

## References

Mohamed Abdalla, Krishnapriya Vishnubhotla, and Saif M. Mohammad. 2023. What makes sentences semantically related: A textual relatedness dataset and empirical study.

Junghoo Cho, Hector Garcia-Molina, and Lawrence Page. 1998. Efficient crawling through url ordering. In *Proceedings of the Seventh International Conference on World Wide Web 7*, WWW7, page 161–172, NLD. Elsevier Science Publishers B. V.

Docsumo. 2024. 7 common data extraction techniques for efficient information retrieval.

Google. 2024. In-depth guide to how google search works.

Ali Pesaranghader, Norwati Mustapha, and Ahmad Pesaranghader. 2013. Applying semantic similarity measures to enhance topic-specific web crawling.

Jacob Wenger. 2018. Six degrees of wikipedia. `https://github.com/jwngr/sdow?tab=readme-ov-file`.

Wikipedia. 2024. Cosine similarity.