Wikipedia Search Engine

Project Proposal

Drew Mink
armink2@illinois.edu

Leo Smat
lsmat2@illinois.edu

Aabha Vyas
aabhav2@illinois.edu

Summary

We plan on creating a search engine for Wikipedia. Our search engine will be written in Python and will scan Wikipedia for relevant pages.

1 Introduction

We reviewed literature relating to information retrieval techniques, natural language processing (NLP) for understanding query intent, and machine learning models for ranking documents. Studies such as "Efficient Crawling Through URL Ordering" by Cho et al. and "The Anatomy of a Large-Scale Hypertextual Web Search Engine" by Brin and Page provide key insights into the challenges and solutions for creating a specialized search engine.

We specifically plan on creating a search engine for Wikipedia. Our search engine will be written in Python and will scan Wikipedia for relevant pages. Our motivation behind this project is to explore the best search and information retrieval techniques that are user-centered effort to enhance how users interact with and retrieve information from Wikipedia. Our goal is to design a search engine that not only increases search result relevance but also improves the overall user experience on Wikipedia. Through this project, we hope to provide a beneficial tool for educational and research purposes, facilitating better access to material in the digital age.

2 Description

The intended goal of our project is to be able to take any user-given query, store it in a retrieval model, parse through Wikipedia documents, and return the most relevant documents to the user based on their query.

We plan to use Python with libraries such as PyTorch to parse the wikipages and construct models for estimating relevance based on a query.

If need be we will use a database such as MongoDB or Postgres to store models detailing term frequency and/or links with keywords.

To process queries and achieve efficient document matching we'll write our own program in Python, potentially explore/use Python libraries to parse our document model database and rank/match the most relevant documents to the query.

We will attempt to parse a list of wikipages from Wikipedia. If we are unable to find such a list, we will construct one as best as possible and use internal links for further searches. Internal links could also be used to help determine relevance of related documents. There is also a TensorFlow dataset of cleaned wikipedia articles that has a broad set of data for us to use.

Use cases include users trying to find wikipages related to their query. This could include students, teachers, or just average users of Wikipedia and the internet.

We will initially use term frequency to rank/match documents to a query, then employ our model on a range of queries and test to see whether they return predetermined most relevant documents. We may explore other ranking/matching algorithms including inverse document frequency or weighting based on document length. We will also experiment with more sophisticated algorithms for improved relevance.

We will also employ relevance testing by evaluating the search engine's performance by measuring how well the returned documents match a predetermined set of relevant articles.

We can add options for feedback from a user to update the models. This feedback loop will allow for users to provide feedback on search results which allows for a continuous improvement of our search algorithm. At a later date we can use different methods for indexing and querying based on new information that we learn in class.