SWE RESEARCH SUMMARY

This document is a brief summary of the core research concepts of the Software Engineering project of our group.

The project itself is meant to be a search engine optimized more specifically for technical documentation, with functionality to add trusted pages rather than searching the entire internet.

This approach provides two notable benefits over searching for technical details on regular online engines:

1. Being able to select explicitly trusted pages (top-level domains and possibly included subdomains) will introduce moderation and will exclude the typical sites advertising their product with content copy-pasted from forums
2. Optimizing the search engine to find technical information in dense text should optimally make it so relevant parts of technical docs can be looked up easily

As part of the challenge and for the purposes of optimization, a major backend component of this project is building a custom search engine from scratch.

In the following sections, the conceptual basis of the custom search engine is outlined.

Page Selection and Processing

Firstly, one of the two main features of the search engine is to limit the scope of the entire internet to a select few pages we mark as trusted.

The primary goal of this functionality is to make sure only websites marked as useful are searched, to make it easier to find the relevant information the user is looking for.

As a side effect, this functionality also allows us to process each webpage in more detail, which should optimally lead to more effective searching. Because of this, we can use more techniques for data search and ordering with the same amount of potentially limited processing power, as we have less total pages to process.

Pages to be processed should be entered manually, optimally by either administrators or verified users. For the version used for this group project, the plan is to index the online copy of the Linux Manual at man7.org, maintained by the maintainer of the actual manual pages. Optimally though, the finished project should be able to add any other websites we choose.

Search Engine Overview

The search engine itself would be composed of three separate components, each using a different method to help search and sort information. All the components

would be unified at the start by a joint preprocessing strategy, and at the end by a way to include all three in the final score calculation.

The word preprocessing component is planned to not be the main focus, therefore, it would consist of the typical, well-described approach of punctuation, stop word removal, word stemming and similar.

The three main parts used for the search itself are planned to be as follows:

- Word Spelling Similarity
- Word Meaning Similarity
- Metadata Search

Each is described in the following sections.

Word Spelling Similarity

The goal of this part of the program is to perform a full-text search, looking up which processed webpages match the words in the query, based on the spelling of said words.

Optimally, the result would be a similarity score, assigned based on how close the words of the document and query are to each other. This means that words with a similar, but not quite the same spelling would still result in decently high similarity values, to account for typos and similar spelling inconsistencies.

The strategy is to be implemented by first converting the words to numbers or vectors (or general N-tensors), in such a way where the distance of the two vectors is directly proportional to the similarity of the words. The exact same words should correspond to the exact same vectors, and words with only a few letters changed should correspond to vectors in close proximity of one another.

This way of searching can be used for full-text fuzzy search in all the relevant documents, which should present decently relevant results even by itself, let alone with the other components.

Word Meaning Similarity

The goal of this part of the program is also to perform a full-text search, however, this time considering the meanings of the words instead.

The result would also be a similarity score, this time assigned based on how close the meaning of the given words match. This is more difficult to implement, as the meaning of a word is not an inherent property that could be derived from just the letters. However, it has been decided that it would be useful in a search engine of this kind, so as to help match results which may be very relevant, just consisting of different words, such as synonyms and paraphrases.

This strategy is to be implemented by converting the words to vectors, using a previously described process known as word embedding. Either just words, or sentences, or entire paragraphs are to be encoded using a word embedding

approach, which will then allow matching queries to documents based on vector similarity/distance.

This approach has a lot of potential weaknesses, but also a lot of potential merit. The current consensus is that this would likely not be reliable enough by itself, but should improve the accuracy when used in conjunction with the spelling similarity score.

Metadata Search

The third part of the engine relies on page metadata, both explicitly specified in the page (considering an html page with 'meta' tags) and data generated from the page.

As the usage of the meta tags over typical webpages seems widely inconsistent, and in general they seem underused, there's two proposed aspects to this component.

Firstly, certain meta tags (such as keywords etc.) as well as the page title, are to be added to the processed text for the rest of the engine to use.

Secondly, generated metadata is to be used to aid in lookups, especially a metric of word frequency (the current options include TF-IDF or simply a normalized term occurrence frequency value, disregarding other documents). Using this metric, the search engine could do a very efficient preliminary score calculation, to see which articles even mention any words in the search query. Since the term frequency values would be generated during the preprocessing phase, this operation would be mainly accessing the values, which is fairly inexpensive. Then, any pages with a score below a certain threshold could be excluded altogether.

This would be helpful in speeding up the search engine, as it would filter the available documents before even attempting lookup using the other two components. The similarity scores would then be calculated for a more limited number of pages, and would then be used to sort the results.

Final Formula

With this approach, the result of looking up a query in a preprocessed database of pages from a given trusted website would lead to a few values:

- The **Word Spelling Similarity** (`Ws`, optimally normalized such that `0 <= Ws <= 1`)
- The **Word Meaning Similarity** (`Wm`, optimally normalized such that `0 <= Wm <= 1`)
- The **Metadata Page Relevance** (`M`, optimally normalized such that `0 <= M <= 1`)

The proposed formula to unite these variables into a final score is akin to a weighted average of the three available values. For a reasonable weighting approach, the following constants are introduced:

- The **Metadata Relevance Constant** (`R, 0 <= R <= 1`)
- The **Meaning Confidence Value** (`Cf, 0 <= Cf <= 1`)

The Metadata Relevance Constant would be an arbitrarily chosen value within the given bounds. The specific value for the most optimal performance is to be figured out using a trial-and-error approach, and the user could also be free to set the value themselves.

The Meaning Confidence Value would be calculated separately for each query, based on how many words in the query exist in the chosen word embedding approach. This is to address the most notable shortcoming of the word meaning approach, that being the fact that no realistically existing embedding can cover all words any user could potentially search for. Potentially, the value could correspond to a sigmoid-type function between a specified lower and upper bound, proportional to the number of words in the query and the number of known words.

Considering all the described variables, the actual score calculation formula could look akin to the following:

```
SCORE = (R * M) + (1 - R) * ( (Cf * Wm) + (1 - Cf) * (Ws) )
```

Resulting in a weighted average of the three values, based on both the relevance constant and the confidence value.

Conclusion

The project requires research in various areas, based on the planned components for the search engine, which is fundamentally what the entire project is about. The search engine is split into three distinct areas, those being a word spelling search, word meaning search and document metadata and term frequency search. All three areas would then be united by a joint formula to calculate a final score for each page, which would be used to order the pages.

All three areas are planned at a high level, with general ideas of what the result should be. There is active research being done by the team on each component, and this is where the bulk of the work for this project would be focused. Optimally, this would also be the core of any potential innovation and research in relation to the project.