Jacob Kickbush

jamakick

I427 Search Informatics

Final Exam

1. Crawling
2. Breadth first search algorithms look through every link on a given page in succession before moving to any of the inner links, whereas depth-first search algorithms go as far into the inner links as they can before it returns to the original page and does it again for each link. Thinking of it as a tree, breadth first goes across all the top branches before moving to the bottom ones, and depth first goes from the top branch to the bottom in a column, before going to the second column of the top branch. Best-first search algorithms try to figure out which link will provide more information or be more suitable to add to your crawler, since you may have a limited amount of pages to look through. For example, a page that has no other links and very little content may not be chosen over a page with 200 links and multiple paragraphs. Breadth first is preferable when you want a broad amount of pages that may have little information on each subject, but has a lot of subjects in it, depth first is preferable when you want a lot of information on each subject that it contains, and best first is preferable when you want the highest quality pages overall.
3. Politeness means respecting the webmasters rules and not overloading the server. It is important because if you disrespect the rules of a website, your crawler may be denied access to it for future scrapings. This concept comes into our search engine because we look at the robots file on the website and ignore the pages the webmaster doesn’t want us scraping. We also limit our requests at a certain amount of time, so we only send a request every couple seconds rather than a million at one time. Freshness means how long it’s been since a certain page has been scraped. This is important because the fresher your pages are, the more likely that you have all the content on them and it has not been updated. This is important to our search engine because a website like CNN.com will be changing all the time and will need to be scraped often to keep it fresh, while a website that is someone’s personal portfolio doesn’t need to be scraped as often, as it may only be updated every couple of years.
4. Indexing
5. First we scraped the urls and used a request module in order to access the html from the page. We then saved all of these pages and their urls to a folder in order to then index them. We indexed it by first getting rid of all the whitespace and unnecessary characters to clean up the html a little bit trying to make it only text. After this we removed all the stopwords that we don’t want to add to our index such as “the”. We then tokenized and stemmed using the porter stemmer every part of the html to generate a list of all the terms that are present in the html page. We then assigned this to a dictionary where the key was the term and the list of all pages containing it was the value.
6. An inverted index is where we have a list of terms that we look through, and the values for those terms show where they are found. This is useful since we can find where every term is found without having to look through every single page to see if they contain that term.
7. Tokenizing is one specific level of information and it is useful because it can break apart different terms into their own parts in order to have a more expansive list of terms. A second is stemming, this looks at the ends and beginnings of words to try to match those that are similar. For example, apple and apples are related to each other but are not the same word, stemming tries to give them commonality.
8. Stopwords are a list of words that are common but not important in text. Prepositions and pronouns are good examples of this. Words like “the” “he” “she” “it” are present in the English language quite a bit, but are not the important parts of the document. By using these and removing them from our index we can create a more valuable list, rather than having a bunch of these useless words always at the top.
9. Retrieving
10. Tf-df is term frequency inverse document frequency. There are several parts to it. First is ntf, which stands for normalized term frequency. It finds the given amount of a word in a page, and then divides it by the total number of words in the page to find a better estimation of the word’s importance on that page. We also have idf which is inverse document frequency. If a word occurs 100 times in a document it doesn’t necessarily mean that word is 100 times more important than other words that only appear once. So we take one and divide it by the log of the term frequency to get a dampened or inverted document frequency. Tf-idf takes these two parts and multiplies them together to get an even stronger measure of importance of words on the page. This is better than raw term frequency because seeing the word potato 1000 times on a page doesn’t mean that page is the best source of information about potatos since it may be all the page has. The same goes for normalized term frequency since you can have a lot of words on your page but they don’t necessarily have to make the term important. By using the different parts together we can get a better estimation of importance with tf-idf.

Section 2

1. SEO
2. Search Engine Optimization is the process where you try to make your page look more attractive to a search engine in order for your results to be put closer to the top. For example, having invalid html compared to having validated html can affect your rankings on a search engine like Google. White hat SEO is where you try to use positive techniques to get your website to the top, such as having valid html, having quality information, and even having a good domain name. Black hat SEO is where you try to take advantage of parts of the system to be pushed towards the top. An example of this is having a website about cats and you want it to be the first result when someone searches cats. So you put the word cats in your website a lot, and then you create tons of different pages that all link to your cat website, trying to cheat the search engine into thinking that your page is important. One example of relevance for each would be for white hat to have a lot of reputable sites link to your page, such as a news source or a company’s website. For black hat a relevance example would be creating 1000s of pages that all link to your crappy site to increase it’s relevance. For trustworthiness, Having valid html helps with white hat SEO and having a page that has a bunch of links to disreputable websites can affect your trustworthiness for black hat SEO.
3. Levenshtein

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | I | N | D | I | A | N | A |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| L | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 |
| O | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 |
| U | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 4 |
| I | 4 | 3 | 4 | 4 | 3 | 2 | 2 | 2 |
| S | 5 | 4 | 4 | 5 | 4 | 3 | 3 | 3 |
| I | 6 | 4 | 5 | 5 | 4 | 4 | 4 | 4 |
| A | 7 | 5 | 5 | 6 | 5 | 4 | 5 | 4 |
| N | 8 | 6 | 5 | 2 | 2 | 2 | 1 | 2 |
| A | 9 | 7 | 6 | 3 | 3 | 2 | 2 | 1 |

1. Recommendation
2. For relevance on movies, I would say the most important features would be the number of views, the date it was added, the rating, the reviews, the user’s viewing history, and possibly the user’s age. Finding the relevancy of the movie itself would be a combination of when it was added to the site, how many views it has, and what reviews the movie has. Weighting these together would be the relevance on the movie on the site, but not the relevance to a user itself. Taking the user’s viewing history, the rating of the movie, and the users age could help determine whether the movie would be relevant to the user or not. You can test this by having a user watch a lot of pg-13 movies, and if a movie has a high relevance and is also pg-13, then it may relevant to the user. The user’s viewing history would factor into this as well, since if they have watched similar movies in the past it would make that movie more relevant as well.
3. Establishing genre is possible with the user data I used in the last example. The user’s viewing history, their age, and the rating on the movie itself can help with this. For example if the user watching it has an age of 18 and they watch movies that are rated M, and in their viewing history they watch movies rated M, you could say that any movie that is rated M and is viewed by people of similar ages may belong to the same genre as the other movies viewed. Taking these separate pieces of data and finding the commonality can provide the same feeling as looking at movies in a genre.
4. Recommending movies based on what similar users have watched can be helpful here. This is collaborative filtering where if one user watched Moana and The Lion King, and another user has watched Moana, chances are they might like the Lion King as well. This is user to user. You can also do item to item where a user would watch a movie like Kill Bill, and they would have Django Unchained recommended to them because the movies are similar to each other. Content based filtering for movies would be to have a list of movies inputted that are similar to the specific movie that you are looking at. For example, every Disney movie would have a list of other Disney movies that the user may want to watch, so the system would recommend this content.