Lime H - Crawling

Chen, Qianjun | Onwuasoanya, Anthony | Li, Danny | Chen, Shuyang

Large-Scale Programming and Testing Project 1 - Search Engine

1. Architectural Divisions and Design
    1.1. Interaction with other two components
        1.1.1. with text transformation team
        send raw htmls, url, access time
        1.1.2. with link analysis team
        Receive the URLs
        Send feedbacks of links (dead links, etc)
        1.1.3. API
        We will provide one POST method for link analysis to send the crawler.

Input	URLs as sources or starting points for crawling websites.
Output	None
Side effect	Links received from link analysis will be added into queue and raw html documents of links that have been crawled will be downloaded with GET request sent to the web and then stored.
Workflow	1. When this method is called, all the links will be passed into crawler. 2. The crawler will give feedback of links, and download raw documents without parsing. 3. The POST request will be sent to link analysis to provide them the Links status. 4. The POST request will be sent to text transformation to provide them the raw html documents for parsing.

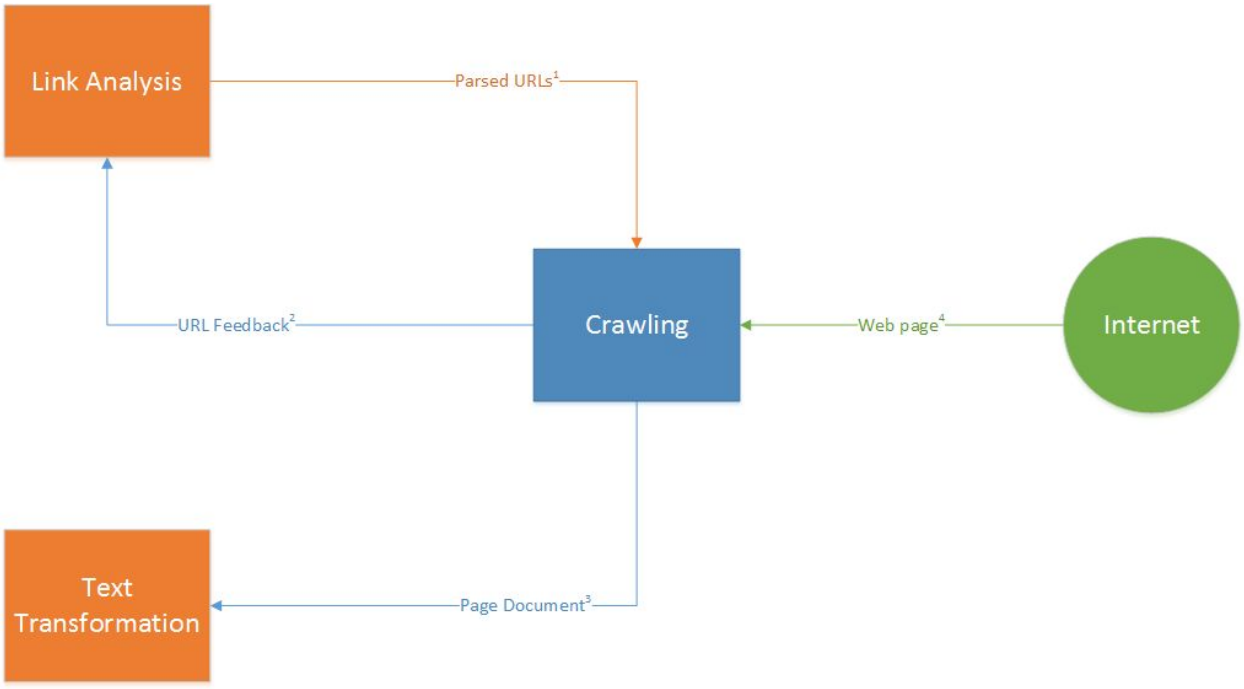
    1.2. High Level Design
        1.2.1. Parsed URLs
        Source: Link Analysis
        Destination: Crawler
        Description: List of URLs to crawl
        Method: POST
        MIME type: application/json
        Response: None
        Action: URL added to queue
        1.2.2. URL Feedback
        Source: Crawler
        Destination: Link Analysis
        Description: List of bad links
        Method: POST
        MIME type: application/json

Response: None
Payload format: {link:reason}
Trigger: Violation of robots.txt or page was not found

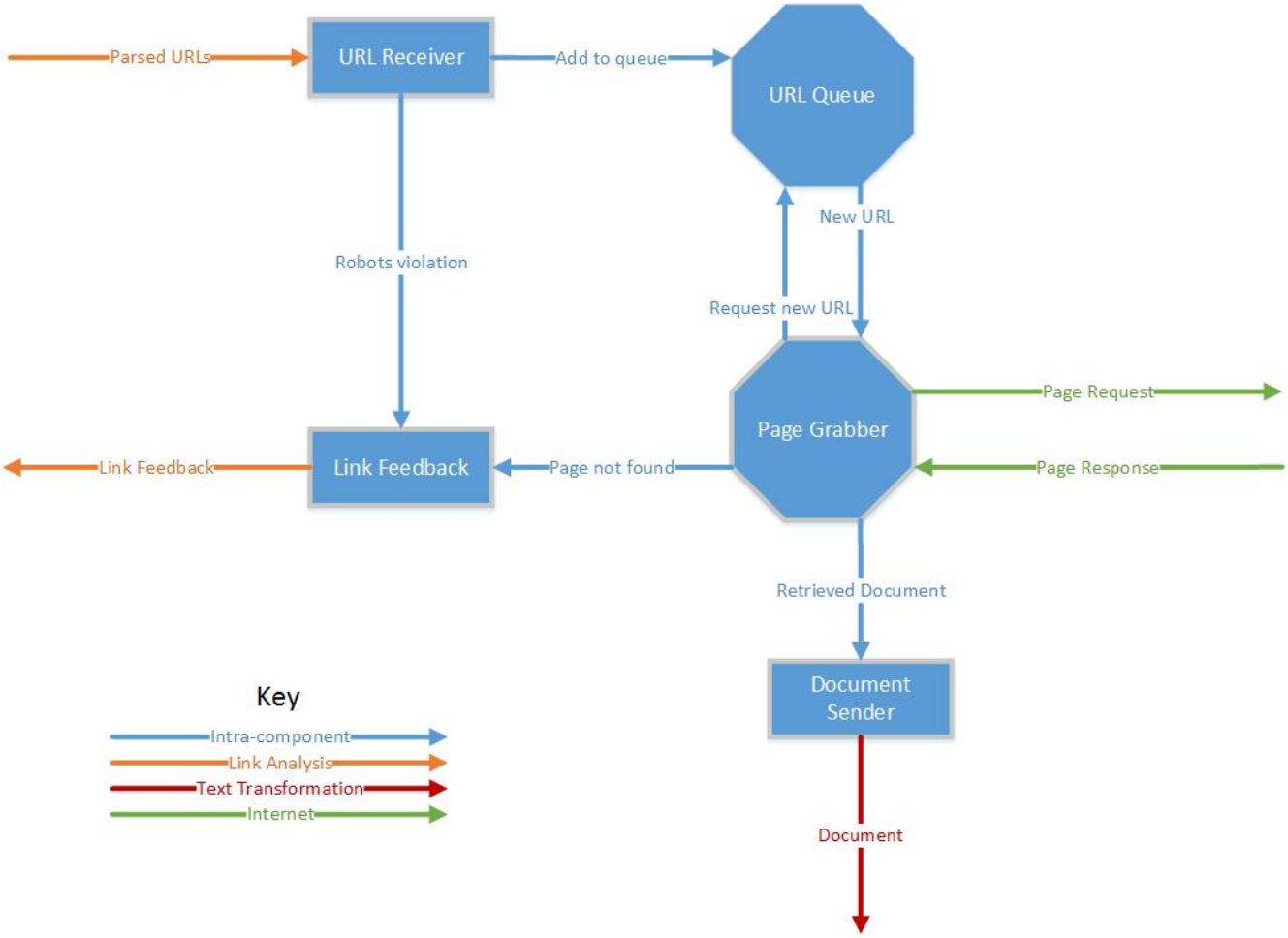
1.2.3. Page Document
Source: Crawler
Destination: Text Transformation
Description:
Method: POST
MIME type: application/json
Response: None
Trigger:

Source: Internet
Destination: Crawler
Description: Download page document
Method: GET
MIME type: text/html
Trigger: URL pulled from queue

Architectural Division



Internal Architecture



- URL Receiver (static): Endpoint to receive links from Link Analysis. Checks for robots.txt violations and passes link to URL Queue or sends to feedback queue in event of violation.
- URL Queue (static): Queue data structure. Stores links sent by URL Receiver. Receives request for new URL from Page Grabber instances and replies with first URL in queue.
- Link Feedback (static): Handles calls to Link Analysis link feedback endpoint.
- Page Grabber: When idle, requests URL from URL Queue. Downloads corresponding page from internet and passes it to Document Sender.
- Document Sender (static): Handles calls to Text Transformation document endpoint

1.3. Responsibilities

1.3.1. Each team member is responsible for their own subset of implementations in the project

1.3.2. Anthony
    1.3.2.1. Software Development (Page Grabber & Document Sender)
1.3.3. Danny
    1.3.3.1. Software Development (URL Receiver & Link Feedback)
1.3.4. Shuyang
    1.3.4.1. Software Tester (Testing & URL Queue)
1.3.5. Qianjun
    1.3.5.1. Software Tester (Testing)

2. Test Plans
    2.1. The overall goal of testing is to find yet-undiscovered bugs and defects. Specifically, in this project, we would like to test the function of acquiring documents for the search engine, as well as the interaction with other components such as text transformation and link analysis.
        2.1.1. Unit Testing
        The goal of unit testing is to test the functionality of crawling: downloading docs online, monitor the memory, adhere to the robots.txt, report bad URLs, etc. We come up with 21 test plans for unit testing (test ID from 1-21 in the spreadsheet).
        2.1.2. System Integration Testing
        The goal of system integration testing is to test the API to other components such as text transformation and link analysis. We come up with 4 test plans for system integration testing (test ID from 22-25 in the spreadsheet).
        2.1.3. Performance Testing
        The goal of performance testing is to test the response rate and scalability of the crawler. We come up with 2 test plans for performance testing (test ID 26 and 27 in the spreadsheet).
        2.1.4. Stress Testing
        The goal of stress testing is the test that determines the robustness of software by testing beyond the limits of normal operation. We come up with 1 test plan for stress testing (test ID 28 in the spreadsheet).
        2.1.5. Test Spreadsheet
        https://docs.google.com/spreadsheets/d/1c3H9tq3cGGdabWBH_5-JzwVOubzIsf0lc1UlMQ7Ty9s/edit?usp=sharing

3. Coding Standards
    3.1. We will be using Python for the crawling component of the search engine.
    3.2. Interpreter: Python 3.7.0
        3.2.1. Because we are using Python as the language for the crawling component, we will be using the PEP8 coding standard.

Classes	Classes will be in UpperCamelCase	class Crawler:
Functions	Function names should be lowercase, with words separated by underscores	def crawl_html:
Variables	Variables will be in all-lowercase with words separated by underscores	spider_name = ‘Charlotte’
Constants	Constants are usually defined on a module level and written in all capital letters with underscores separating words.	REDIRECT_ENABLED = False
Modules	Modules should have short, all-lowercase names. Underscores can be used in the module name between words.	import scrapy

        3.2.2. Our code will adhere to the following
            3.2.2.1. 79 character line limit
            3.2.2.2. Single quote strings
            3.2.2.3. No extraneous white spaces
            3.2.2.4. Surround top-level function and class definitions with two blank lines