algorithms

MDPI

Article

Building a Technology Recommender System Using Web Crawling and Natural Language Processing Technology

Nathalie Campos Macias, Wilhelm Düggelin, Yesim Ruf and Thomas Hanne *

Institute for Information Systems, University of Applied Sciences and Arts Northwestern Switzerland, 4600 Olten, Switzerland; nathalie.campos@students.fhnw.ch (N.C.M.); wilhelm.dueggelin@students.fhnw.ch (W.D.); yesim.ruf@students.fhnw.ch (Y.R.)
* Correspondence: thomas.hanne@fhnw.ch

Abstract: Finding, retrieving, and processing information on technology from the Internet can be a tedious task. This article investigates if technological concepts such as web crawling and natural language processing are suitable means for knowledge discovery from unstructured information and the development of a technology recommender system by developing a prototype of such a system. It also analyzes how well the resulting prototype performs in regard to effectivity and efficiency. The research strategy based on design science research consists of four stages: (1) Awareness generation; (2) suggestion of a solution considering the information retrieval process; (3) development of an artefact in the form of a Python computer program; and (4) evaluation of the prototype within the scope of a comparative experiment. The evaluation yields that the prototype is highly efficient in retrieving basic and rather random extractive text summaries from websites that include the desired search terms. However, the effectivity, measured by the quality of results is unsatisfactory due to the aforementioned random arrangement of extracted sentences within the resulting summaries. It is found that natural language processing and web crawling are indeed suitable technologies for such a program whilst the use of additional technology/concepts would add significant value for a potential user. Several areas for incremental improvement of the prototype are identified.

Keywords: recommender systems; web crawling; natural language processing

Citation: Campos Macias, N.; Düggelin, W.; Ruf, Y.; Hanne, T. Building a Technology Recommender System Using Web Crawling and Natural Language Processing Technology. *Algorithms* **2022**, *15*, 272. https://doi.org/10.3390/a15080272

Academic Editors: Asit Kumar Das and Danilo Pelusi

Received: 13 June 2022
Accepted: 29 July 2022
Published: 3 August 2022

1. Introduction

Artificial intelligence and more specifically the combination of web crawling and natural language processing have been used to build various types of recommender tools in previous research as well as in development projects. Our research project aims to find out whether the combination of web crawling, web scrapping, and automatic text summarization with natural language processing technology can result in a prototype which is adequate for yielding recommendations on trending technology for a selected application purpose to a subject matter expert. We show how the mentioned technologies can be used in combination to provide a suitable solution for automatic information extraction and knowledge discovery in the considered application context and show the potential benefits by evaluation experiments.

1.1. Problem Statement

In today's environment of high-speed development and complex markets, it is difficult even for experts in a particular domain to keep track of recent technologies. Furthermore, once information on a technology is found, it is difficult, and time consuming to understand and evaluate it roughly and decide whether it deserves a more detailed investigation.

Our research investigates the possibility to automatize the task of keeping track with recent technologies such as hardware, software, or other types of engineering solutions, with the help of web crawlers in combination with algorithms related to natural language processing.

1.2. Thesis Statement and Research Questions

The following hypothesis and research questions were derived from the problem described in the problem statement.

Hypothesis: Web crawling and natural language processing technologies can make the search for and basic understanding of domain specific technology more efficient and effective than the use of popular online search engines.

Independently of the results regarding our hypothesis, the following research questions will be answered to support a potential falsification or acceptance of the hypothesis.

RQ1—Feasibility: Is it possible to build a prototype of a technology recommender system based on natural language processing and web crawling technology using programs in Python?

RQ2—Effectivity: Can such a technology recommender system yield satisfactory results in terms of the quality of recommendations? Quality means that the recommendations are easy to understand and suited to the selected application field.

RQ3—Efficiency: Is the quantity of results satisfactory? By "satisfactory" we mean that the recommender system delivers at least the same number of results as a manual web search including a suitable summarization but in a more time-efficient manner?

1.3. Research Approach

Based on the hypothesis and research questions RQ2 and RQ3, we follow a design science research approach to develop a prototype which is compared with manual search regarding efficiency and effectivity in the evaluation stage of this work. The category of effectivity addresses the quality of results generated by the prototype compared with results generated by manual search, using the Google search engine. Criteria within this category are the fit of the found URLs with the search terms/keywords, the subjectively perceived usefulness of the generated text summaries, and the number of potential duplicates. Within the analytical category of efficiency, the only criterion is the time used to find and retrieve relevant information.

1.4. Scope and Limitations

The development project underlying this research paper uses opensource libraries wherever possible and further makes use of modules which allow the use of the Google search engine. The use of artificial intelligence within the developed artefact/program is limited to the application of natural language processing. Evaluation experiments were conducted as self-experiments through the authors and any results should be viewed as explorative only. The human use of the Google search engine will serve as a unique tool for comparison and evaluation which further limits the scope of this paper. Due to the limited scope of this work, any conclusions drawn from the hypothesis cannot definitely confirm or deny the possibility of the underlying idea.

2. Literature Review

The following sections aim to summarize relevant literature from academic research, as well as academic and commercial development projects with the aim of providing the theoretical basis for this project as well as determining the state-of-the-art and consequently any relevant research gaps.

2.1. Related Work

When screening the academic literature, it becomes evident that the problem of finding adequate information in the web is widespread. The papers from Wang et al. [1] and Lee et al. [2] explain methods how web crawlers using key words can be used for recommender systems in the field of scientific journals. Both papers explain ways how to extract content from documents with natural language processing (NLP) technology and find adequate results in the web by using web crawling methods and recommend similar scientific papers [1,2]. A search query on Scopus revealed 431 results when entering the

combined key words: "web crawling" AND "natural language processing". To reduce the results, the filter on document type were set to book, book chapter or article and the subject area was limited to computer science. Thereby the results were reduced to around 150. Many of the results explain the technique of web crawlers, where crawling combined with natural language processing helps analyze big data in the web. Similar attempts are made in many different fields. For example, Rajiv and Navaneethan [3] optimized the web crawling with a keyword weight optimization in an event focused web crawling. The work of Vural, Cambazoglu and Karagoz [4] explains how sentiments can be crawled by designing a sentiment-focused web crawling framework. Bifulco, Cirillo, Esposito, Guadagni and Polese [5] released a document presenting an intelligent system which is tested especially in the field of e-procurement to support organizations in the focused crawling of artefacts (such as calls for tender, equipment, policies, market trends, and so on) of interest from the web, semantically matching them against internal big data and knowledge sources [5]. A comparable document is also found in the field of ethnopharmacology written by Axiotis, Kontogiannis, Kalpoutzaki and Giannakopoulos [6]. Alarte and Silva [7] discuss the problem of only extracting the relevant content from a website, i.e., ignoring content such as menus, advertisements, copyright notices, and comments. They suggest an effective method for this purpose. A broad survey on web data extraction techniques and applications is provided by Ferrara et al. [8]. This paper illustrates well the broadness of use cases and application domains of the technology. A concise overview of web data extraction including scientific background and some tools is provided by Baumgartner, Gatterbauer, and Gottlob [9].

As mentioned for the Special Issue edited by Angulo et al. [10], future trends in that area may be based on a better integration of recommendation systems with cognitive approaches which should improve cognitive reasoning. Aspects of personalization and personality-aware recommendation systems [11] might also be promising for systems based on web content extraction. However, as our considered use case assumes more or less anonymous users, we do not delve further into this field.

All the mentioned research on web crawling and natural language processing aims to find methods that help reduce manual and time-consuming effort by searching the Internet for information in different fields of interest and mostly based on own known data.

For our focus, a recommender system for technology based on an open web search, we combined the key words "recommender system" with "NLP" and "web crawling". This search resulted in only 17 papers. When looking up "recommender system" and "web crawling" a paper similar to the intended research topic came up. The article from 2019 [12] explains how to use effective web scraping methodologies, where the data are initially extracted from websites, then transformed into a structured form.

2.2. Research Gap

As a results of analyzing the literature on NLP, web crawling, and recommender systems it became evident that most research is concerned with how data are available compared to websites and gives a benefit to a user by finding the best match to this available data. In the field of product recommendations or job recommendations the technology is well-known and used. The before mentioned papers [1,2,5] are examples of research using the technologies in more specific fields. But also in those examples, data are available, and NLP is used to find the right search terms for the web scraping to recommend suitable results.

Reports of projects that combine web crawling and the recommendation of recent technologies without the need of substantial data as an input, were not found during the literature research.

An output of summarized search results in text form was not mentioned in any of the articles found during the literature research. Text summarization is nothing new but apparently it is not used combined with web crawling in recommender systems. A research gap can be filled by developing and testing a technology recommender capable of finding

information according to only few input criteria, across different online sources, and automatically arranging the resulting information in summaries.

3. Research Design

The first two subsections of this section describe the research methods considered as well as the research strategy and approach adopted, including the design of the evaluation experiment. The subsequent subsections cover the technological concepts considered for the development of the artefact.

3.1. Research Methods

In information systems research, two research methods are of special significance: design science and action research. Carstensen and Bernhard [13] categorize design science research as a qualitative research approach. According to Hevner and Chatterjee [14], design science research can be defined as a paradigm in which designers answer questions which are relevant to the problems of humans whilst simultaneously providing new scientific evidence so that the artefacts designed are both useful and of fundamental nature to understand the addressed problem. Hevner and Chatterjee [14] also suggest that a first principle of design science research is that the understanding and knowledge of a problem and its solution are best acquired in developing and applying an artefact that acts as a remedy to the problem.

Other common frameworks for research strategies in information systems research are case study and action research. Action research can be defined as a cognitive process, depending on social interaction between observers and people in their surroundings [15]. As stated by Blum [16], action research refers to the diagnosis of a social problem with the intent of improving the situation. Action research has two stages: The diagnostic stage, where the problem is analyzed and hypotheses are developed as well as a therapeutic stage, where the hypotheses are tested by experimenting with changes within the respective social system [16].

3.2. Chosen Approach and Research Strategy

For this work, a qualitative research method was chosen for reasons of simplicity and its suitability to the chosen research strategy which can be defined as simplified design science research, adapted from [14]. Design science research was deemed more suitable than action research since the latter focuses more on observing the impact of experimental changes to an existing social system e.g., by changing a process or an information system used within the context of a social system. Further, the action research aspects of stimulus and reaction are not given within the context of the artefact developed and evaluated within the scope of this work.

Our research strategy differs from what has been suggested by Hevner et al. [17] as the process showed in Figure 1 does not include the development and evaluation of design alternatives.

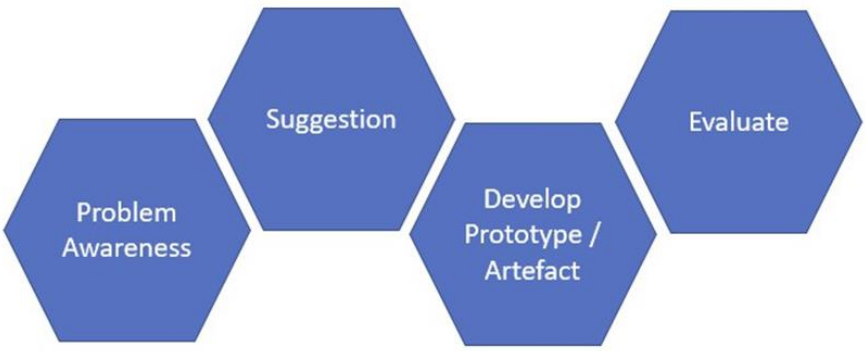

Figure 1. Research strategy based on design science research as suggested in [14].

The research strategy underlying this work starts by becoming aware of the given problem. The awareness of the problem was identified prior to this work and is further elaborated in the introduction and literature review of this paper (see Section 2). The start of the suggestion stage was marked by a workshop conducted by the authors to lay out the established process underlying the research problem. A first solution idea was then suggested before the artefact was developed within the scope of an executable prototype. The prototype development was iterative and involved incremental testing, adjustment, and extension. In the last stage, the prototype underwent a final qualitative evaluation within a self-experiment conducted by the authors. To evaluate the prototype, the process and results of a conventional Google search by a human were compared with the process and results using the prototype.

The tests aim to measure the relative effectivity and efficiency of the prototype. They make it possible to give answers to the research questions formulated in Section 1.2.

3.3. Relevant Technological Concepts

The following subsections explore technological concepts that support the development of a prototype that can support Internet search for technology by a human.

3.3.1. Natural Language Processing

According to IBM [18], Natural Language Processing (NLP) is the domain of artificial intelligence (AI) which enables computers to understand written and spoken words in a way that approximates human understandings of such words.

3.3.2. NLP for Text Summarization

There are two methods of automatic text summarization. One is the extraction-based method of text summarization, and the other is the abstraction-based method. The differences according to Garbade [19] are:

- Extraction-based or extractive text summarization involves extracting key phrases from the source and combining them to create a summary. Such extraction is done according to pre-defined metrics and without any alterations of the text.
- Abstraction-based text summarization includes the paraphrasing and shortening of elements within the source. Abstraction is used in text summarization in the context of deep learning problems and has the capability of overcoming grammar inconsistencies which are often associated with extractive methods. Abstractive text summarization algorithms create entirely new sentences which include the most relevant information.

For its simplicity, extraction-based summarization was used within this work to develop a first prototype. It is assumed, however, that a future product based on our work could benefit from the use of abstractive summarization.

3.3.3. NLP with Python

Python is a practical programming language, which is freely available. Python is dominating in NLP. It provides extensive libraries and frameworks. Furthermore, Python can be installed on many operating systems. Python's straightforward structure makes it the perfect choice as a first programming language [20]. It should be mentioned that compared to some other programming languages such as C++, Python might show performance disadvantages as it is not compiler-based. While C++ is statically typed, Python is a dynamically typed language due to its interpreter-based concept. It can also be assumed that this Python concept supports faster prototyping and a more interactive software development which is usually desired in fields such as NLP or machine learning. The programming language R with a strong focus on statistical computing might also be a suitable choice for NLP related projects as it includes various NLP related libraries. However, it is still less popular than Python and has a relatively complex syntax.

The before mentioned advantages as well as other benefits such as ease of use, reliability, and the availability of third-party installers made us choose Python as our programming language for this project.

To be able to work with human language data in Python, a specific set of digital tools is needed. The Natural Language Toolkit (NLTK) is a leading platform which provides different corpora, lexical resources, and libraries for text classification, parsing, tokenization, and semantic reasoning [21].

3.3.4. Text-Based Recommendation Using NLP

NLP is being used and has been used for text-based recommendation, e.g., Wang et al. [1] have built a system which helps researchers to decide to which journals they should send their manuscripts based on past publications of the journals. Wang et al. [1] tested a hybrid model based on chi-square feature selection and softmax regression, yielding an accuracy of 61.37%.

3.3.5. Web Crawling

Web crawling and web scraping are important tools to process large amounts of digital text files. Both tools can be integrated into programming environments such as Python or R [22], p. 77. The automatic search of the Internet for specific information and data by a computer program is called web crawling. This is done by web crawlers, also known as bots or spiders. In this process of web crawling, algorithms are used to narrow down the search results to the specific information [23].

According to Ignatow and Mihalcea [22], pp. 77–78, the web crawling process can be summarized in the following steps:

1. The web crawler is annotated with a manually selected list of URLs (seeds). This list of URLs then grows iteratively.
2. The crawler selects an URL from this list. The selected URL is marked as "crawled". Then the crawler extracts further links and content from the selected website.
3. Content that has already been viewed is discarded. The remaining content will be added to the collection of webpages to be indexed and/or classified.
4. To ensure that the URL has not yet been seen and that the page exists and can be crawled, a check is performed for each new URL in the group. Only after all these checks have been passed is the new URL added to the URL list in step 1 and the crawler continues with step 2.

Dilmegani [24] explains that all search engines have web crawlers. Some well-known examples are:

- Googlebot for Google;
- Amazonbot is an Amazon web crawler for web content identification and backlink discovery;
- Bingbot for Bing search engine by Microsoft;
- DuckDuckBot for DuckDuckGo;

The challenge in web crawling is that a web page overload must be avoided and at the same time large amounts of data must be processed. To reduce the amount of data to be processed, the order in which the URLs are scanned must be carefully decided [25].

3.3.6. Web Scraping

According to vanden Broucke and Baesens [26], p. 3, web scraping can be defined as "the construction of an agent to download, parse and organize data from the web in an automated manner".

As explained by De and Sirisuriya [27], with web scraping, unstructured data can be extracted from websites and converted into structured data, which then can be stored in databases and analyzed. Web scraping is a technique of data mining. The web scraping process aims to return the extracted information in an understandable structure such as

CSV files, spreadsheets, or databases. Based on such targeted information extraction, it is easier for businesses to make decisions [27].

According to De and Sirisuriya [27], web scraping is mainly used in the following fields of application:

- For comparison;
- To detect website changes;
- For market analysis;
- For research analytics;
- For contact scraping;
- For job scraping.

3.3.7. Web Crawling and Web Scraping Tools for Python

As some of the authors have little programming experience, the following criteria were crucial for the selection of the tools:

- Should run on Python;
- Easy to grasp and learn;
- Well documented;
- High community support;
- If possible, web crawling, web scraping and Natural Language Processing in the tool should be expandable according to the needs (libraries);
- Built in data storage;
- Can manage complex scraping operations in a powerful manner.

Based on the above criteria, the authors prepared the decision table shown in Table 1 as a basis for their decision-making.

Table 1. Decision table—overview of web scraping and web crawling libraries.

	Scrapy	**Mechanical Soup**	**PySpider**	**Beautiful Soup**
What is it?	web scraping framework	library	web crawler	library
Purpose	web crawler and collaborative web scraping	to simulate human interaction with websites	web crawler	data parser
Use cases	data extraction from websites, processes them as needed, storage in a preferred format	automated storing and sending of cookies, redirections following, link following and form submitting, easy filling in of HTML forms	ability to retry failed pages, crawling pages by age, prioritization, task manager, easy to use web user interface with script editor	screen-scraping, pulling data out of HTML and XML files, navigating, searching, and modifying parse tree
Ideal for	high-level web crawling & scraping projects on a large scale	simulating human behavior	powerful web crawling system	simple web scraping tasks
Speed	very fast	fast with scraping simple websites	fast	fast
Learning simplicity	easy	easy	very easy	very easy
Built-in Data Storage Support	Json, XML, CSV	own development needed	supports MongoDB and MySQL	own development needed
Documentation	excellent	available, unmaintained for several years	very well	excellent

Table 1. *Cont.*

	Scrapy	Mechanical Soup	PySpider	Beautiful Soup
Community support	very high	unmaintained for several years years	high	high
Suitability for the project	excellent	poor	very good	very good

Based on the content from Table 1, we decided to work with BeautifulSoup, because the documentation of Scrapy is not beginner friendly. The following points were crucial for the decision:

- Runs on Python;
- Can be learned with a little more effort;
- Is well documented and beginner friendly;
- Has a high community support;
- Is suitable for web crawling and web scraping, is easily extensible;

4. Suggestion, Development, and Artefact

This section describes how the solution design evolved, how the prototype was developed, and how the solution/artefact works.

4.1. Suggestion and Solution Design

A workshop was conducted to figure out what exactly are the needs that the prototype should be able to cover. The following manual process should be partially automatized by the program:

1. Define search terms;
2. Go to a search engine;
3. Enter search terms;
4. Watch search results and estimate relevance by screening meta description and order of results;
5. Access relevant results and find relevant information on a suitable technology;
6. Decide on important criteria for the suitable technology;
7. Estimate whether the information on the website is helpful;
8. If yes, extract the necessary information by saving the URL and possibly summarizing its key content;
9. Find the next relevant search result and repeat steps 5–8.

Once the manual process was defined, a first idea of the same process with a recommender system was generated:

- Start program;
- Input a search term/filter (e.g., "CRM");
- Start web crawling (e.g., find URLs with the term "CRM" and save them);
- Start web scraping (scan every found URL) and return for example the words that were mentioned the most → Basic text mining;
- Use an NLP technique to filter the results from web scraping;
- Return (e.g., top 10) results in extractive summaries.

A flow diagram of the suggested solution based on existing technologies is shown in Figure 2. Further details of used tools and the overall code script are provided in Section 4.2.

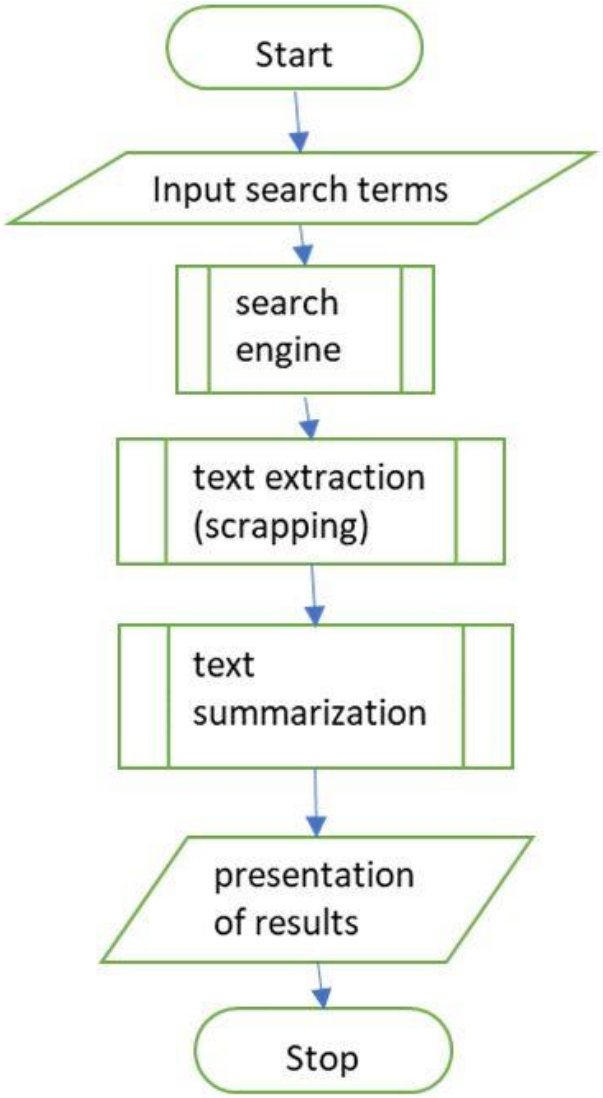

Figure 2. Flow chart diagram of the basic tool chain suggested.

4.2. Development and Resulting Artefact/Prototype

For development we restricted ourselves to the exclusive use of the Python programming language as documented [28]. We also used the Anaconda 3 Integrated Development Environment as provided by Anaconda Inc., Austin, TX, USA [29] and its graphic user interface, Anaconda Navigator, also provided by Anaconda Inc. [29]. As an editor, the authors used Visual Studio Code as provided and documented by Microsoft, Redmond, WA, USA [30].

The resulting prototype artefact does not entail a graphic user interface. The open-source technology used as well as the mechanics of the prototype can be summarized as follows:

Our prototype is based on a number of libraries, packages, and bundles. Before any packages can be imported, they must be installed. This generic task will not be described in detail. A list of libraries, packages, and bundles used is included in Appendix A.

The execute code script is presented in Appendix B. In our prototype, URLs matching the user's inputs are first found with Google search before their content is scraped with BeautifulSoup4 and then summarized extractively using TextRank. The typical runtime for executing this tool chain for the ten most relevant web sites is ten seconds (also cf. Section 5.2).

5. Evaluation

In the last stage of design science research, the evaluation of the artefact is described. As mentioned in Section 3, the evaluation was carried out in the form of user tests that compared the conventional Google search by a human with the prototype's process and results. In this section, the testing is described, and the test results are revealed.

5.1. Test Methods

The setting of the prototype variables "key words", "number of results" and "domain" were decided upon by the test person. In total, five different search queries were performed.

The key words for each query were chosen based on the professional experience of the test person. In order to make an evaluation as comprehensive as possible, different fields of technology were explored. The following key words were used per query:

- New multifunctional device for SME;
- Recommend ERP system for small business;
- Trends ERP system for small business;
- Mixed reality for workshops;
- Business process management tools trends.

The top-level domain was set to ".com". For the first query, the number of results were limited to the default value of five. Especially for the quality test of the text summarization part, it was important to see a broad number of results and therefore the other four queries were extended to a value of ten for the number of results. The number of checked websites was even further increased in two cases as some webpages do not allow web scrapping and the protype has to skip those websites. In "trends ERP system for small business" the number was increased to 11 results and in "recommend ERP system for small business" the number was increased to twenty results. This allowed the prototype to output a total of ten results for both queries.

The human performer executed the following steps for comparison:

1. Go to search engine www.google.com;
2. Enter search terms (key words);
3. Access the top result;
4. Extract URL to result report;
5. Find and extract relevant information to the result report;
6. Access the next result;
7. Repeat 4–6 until the desired number of results is achieved.

Once the queries from the human and from the prototype were finished, both results were added to the evaluation sheets.

In the evaluation sheet A, the functionality of the prototype and the quality of the URL were rated. The following questions were asked:

- Does the prototype extract the URL it is supposed to?
- Does it adhere to the given parameters?
- In the evaluation sheet B, the quality of the text results was rated. The following questions were asked:
- Is the summarized text understandable for a human reader?
- Does the text summarization show the content of the website?
- Is the content informative? "Informative" means that the software provides an answer to the keyword search, and it does not mislead.

Duplicated results were eliminated in evaluation B.

Evaluation C aimed at the time efficiency. A new query was set up for this evaluation because the reading time of the human performer should not be affected by former tests. For this test, the variables were set "domain = com", "number of results = 10" and "keywords = document management system for manufacturing industry". The human performer executes steps 1–5. The following question should be answered with this test:

- How long does each method need from start to the end?

For both methods, the time was taken from the beginning of each process, i.e., for the human when starting the search engine and for the prototype when starting the program. The time was stopped when the defined number of results was scrapped and summarized. The prototype put a result page out and the human performer added their results in the result report. The speed test shows only the efficiency in terms of time, the search results and the quality of the results are not measured in this part of the evaluation.

5.2. Evaluation Results

In evaluation A, the general functionality of the prototype was tested. It was observed that the prototype was able to perform what was expected. The variable "number of results" was correctly considered. As mentioned before, web scrapping is not allowed on every page, therefore the prototype skipped such web pages, and the number of output files was reduced by the count of such web pages. The variable of "domain" was not taken into consideration. The prototype chose the search result from Google without the limitation in the domain.

For each query, the output revealed a result xml-file with the considered URL and the summarized text. The total amount of results aimed was 45 (four queries with ten results and one query with five results). Table 2 shows the results of the comparison between the human search results and the results from the prototype.

Table 2. Results evaluation A.

Result Evaluation A	**No. of Results**
Identical	17
Appears in manual search top 10	12
Duplicated	7
Wrong domain	5
Missing result because of scraping error	2
Did not appear in manual search	1
Subpage	1
Total	**45**

Identical means that the human performer and the prototype found the same URL and listed it at the same position of the search. This outcome was found 17 times. Twelve times the same URL was found by the prototype and the human performer but on a different search result position. Nevertheless, they were among the top ten search results in both methods.

Seven results were duplicated by the prototype. Meaning there was only one URL listed in the Google search results, but the prototype found different URL from the same page and listed all of them as results. As mentioned before, the limitation to the domain "com" was not taken into consideration and five pages were shown with different domains. In two queries, one result was missing due to web scrapping not being allowed on one page. For one website, the Google search displayed a subpage, whereas the prototype retrieved its information from the homepage. The human performer skipped this website whereas the prototype took it into consideration.

In evaluation B, the quality of the text summarization was analyzed. In total 34 unique URLs resulted into text summaries. Table 3 shows the results to the evaluation questions. The first question was whether the summarized text was understandable. In the tester's opinion, 23 results were understandable. Twenty times the content of the summarized text matched the content in the webpage. But only in nine cases the reader had the impression of

receiving valuable information from the text summary. Eleven results matched the content but were still perceived as uninformative.

Table 3. Results evaluation B.

	Summarization Understandable?	Match Content?	Informative?	Match Content and Informative?
No	11	14	35	11
Yes	23	20	9	9
Total	34	34	34	20

Looking at the cases where the summaries were not informative but matching the content gives more insights into the prototype's performance. In Table 4, reasons for the perceived lack of information within those summaries are given.

Table 4. Result evaluation B observations (match with content = yes, informative = no).

Result Evaluation B	No. of Results
Doesn't get to the point	1
Is misleading at the beginning/sense of the text is mixed up	1
Only marketing	1
Products are not mentioned	1
Products are not mentioned, does not get to the point	1
Table with products is not mentioned	1
Website not relevant	3
Website not relevant, summarization failed	2
Total	**11**

In 5 of 11 times, the website was not relevant from the beginning, suggesting that the website would have been skipped by the human tester. Three times the products were not mentioned, even though the queries were aimed for products. The remaining three unsatisfying results showed unnecessary or misleading content from the website.

Next the results from evaluation C are shown in Table 5. It displays the efficiency of the prototype in terms of search time:

Table 5. Results of evaluation C regarding time efficiency.

	Human	Prototype
Human input at process start		25 s
Time to find ten pages	16 min	10 s
Total time	16 min	35 s

Compared to a human using only the Google search engine, the use of the prototype allowed for much faster retrieval of information. However, prototype results were not screened by a human, meaning that during the 35 s of execution no information was processed by the human mind. In contrast, during the 16 min in which the human performed the process using only the search engine, information from ten different web pages was already processed intellectually to some degree.

6. Findings and Discussion

Several observations were made during the tests. The tests reveal that the prototype generally works in terms of functionality. It does that for which it is programmed. Only the

random occurrence of non-compliance with the specified set top level domain requirement was identified as a possible flaw in the functionality.

In terms of effectivity, the tests showed that the information received from the prototype is limited in quality and that the Google search performed by the human was more insightful. The human expectation was to receive some type of technology or product recommendations but in the text summary the product information was mostly missing. Nevertheless, the prototype showed in each query some trending solutions and provided insights into criteria that need to be considered when looking for a specific technology solution. General information about a topic can be retrieved with the prototype, but specialized product recommendations get lost or confused in the text summary. Thus, the product name is mentioned but the text around it does not necessarily have to relate to the product. Some keywords led to better quality summaries than others.

It is subject to interpretation whether the prototype artefact developed throughout this work can be called a recommender system or not. Researchers such as Wang et al. [1] have applied a more stringent interpretation to the recommending aspect of such a system. They propose that business users could use a similar system to that which the authors developed with the aim of simplifying the search for scientific publications where the input would be a summary of what the business user is looking for [1]. In the case of this project, however, the input were short search terms which led to the result that only users who already know the name of what they are looking for can apply the program. For example, if the input were a statement describing the user's problem to be solved, more technical and maybe more unexpected novel results could be found.

7. Conclusions and Outlook

During this project, we essentially developed a program for assisted search engine use with the additional value of generating short, extractive summaries of search results in the context of recommending technologies based on keywords. The design science research method has the objective of yielding both knowledge as a contribution to science as well as a problem-solving artefact that can be used in a day-to-day task [14]. In our case, we managed to develop an artefact that if further improved can be of help for a user who must retrieve larger amounts of summarized information from the web. In this way, our study indicates the potential of automatic knowledge discovery from the large amounts of unstructured information found in the web by means of natural language processing and artificial intelligence-based algorithms. Techniques from web crawling and scraping can effectively search and extract unstructured text information which can further be processed by text summarization algorithms in the context of recommender systems applications.

Our research question RQ1 can be answered with yes as our research of secondary sources during our literature review, together with the insights we have gained into developing such systems throughout the development stage of this project, have shown that the technology is available and that collective knowledge on the relevant topics is vast. However, it must be clarified that in the end, despite finding it possible, no technology recommender system in the truest sense of the word has been developed by us. This is also due to the reason that our solution is very generic and is not specialized for the search of technology.

Similarly, in view of our own solution, research question RQ2 (effectivity) would have to be answered with a no since the results yielded by the program were not of satisfactory quality, mainly due to the reason of a relatively random extraction of text from the content source with sentences that are complete but not necessarily arranged in a meaningful order. Another indicator of low-quality results was the fact that some of the summaries did not match the search terms very well. However, as to effectivity and usefulness, our artefact delivers value for one newly identified use case: since it extracts summaries from websites that use the search terms in any possible way, the screening of summaries that are generated by our program offers the user a unique way to find out in which contexts the search term is also used, so that a user can efficiently find complementary products and services as well

as substitutes. It could be argued however that this can also be achieved by performing a simple Google search which however has the disadvantage of requiring a user to screen each website since the meta descriptions displayed by Google on the search results list contain less information than our extractive summaries.

Research question RQ3 can be answered positively, as the time to retrieve summarized information from the web could be considerably reduced with our artefact if compared to the execution of the manual process using only the Google search engine. Finally, we can partially approve our hypothesis as the technologies of web crawling, and natural language processing, including web scraping have been found to be suitable for development projects that aim at simplifying the search for and understanding of technology. With regards to effectivity however, additional technology such as machine learning and/or advanced statistical methods within and/or outside the scope of NLP would have to be considered in the development of such an artefact.

In order to fulfil the expectations a user could have toward a technology recommender system, further programming and research would be required. The following future work could provide additional benefits to users. For example, the replacement of extractive summarization used in our artefact with an abstractive approach could deliver excellent value as it would yield results with more wholesome information. A further incremental improvement could be the integration of a test for similarity to avoid duplicated results. This could be implemented with genism, a Python library developed by Rehurek [31]. In order to arrive at a user-friendly program, the development of a graphic user interface should also be considered, and efforts toward the compatibility with other content types such as online documents in PDF format would lead to more diverse results. Another suggestion from the authors is the integration of automated translation into several other languages, for example Chinese. If the system were able to translate the input, sources in foreign languages could be found and if the summaries were automatically translated back into English, information about technology novelties from other parts of the world would become much more accessible.

Besides such incremental improvements, future work could focus on developing a technology recommender system which takes a problem statement or even a job description and additionally some content categories such as for example “research”, “commercial products and services”, “tutorials”, “technological concepts”, or “patents”. Each category could then be attributed to specific keywords which would have to appear in a certain density within the online source. The output could then be URLs, abstractive summaries, as well as high density non-stop words found in the source. An ideal program would then run searches in the background, have awareness of existing information and alert the user about new information found. A logging mechanism could be used to keep track of what the user selects for a closer review, allowing a machine learning algorithm such as k-nearest neighbors to learn about the patterns within content that the user deems relevant in order to improve future suggestions.

In addition, it would be useful to automatically generate further information from the extracted content such as semantic information or build-up of a knowledge graph. This may help a user better understand the individual recommendations and to relate them to each other. Furthermore, this concept may also improve the trustworthiness and explainability of the found recommendations. There are still various problems related to the respective approaches such as entity disambiguation, managing changing knowledge, the requirement of manual work, and performance issues.

Regarding the evaluation of the respective techniques, it certainly makes sense to conduct more thorough experiments with users than was possible in our project. This should also include a larger group of participants in the experiments, which could enable a better statistical analysis of the results.

Author Contributions: Methodology, software, validation, writing, N.C.M., W.D. and Y.R.; supervision, review and editing, T.H. All authors have read and agreed to the published version of the manuscript.

Funding: This publication has been partially supported by Innosuisse Innovation project 50775.1 IP-SBM.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Further details on the conducted experiments and related data can be obtained from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Libraries, Packages, and Bundles Used

The following open-source libraries, packages and bundles were imported at the beginning of the script as subsequently explained:

- **googlesearch** as released by Vikramaditya [32] is a Python library that allows Python programs to perform Google searches using requests and BeautifulSoup4 to scrape Google results.
- **nltk.sem.evaluate** is an NLTK (Natural Language Tool Kit) module developed by Klein [33] which provides data structures for representing first-order models.
- **requests** is a HTTP library provided by Reitz [34] which allows software written in Python to send HTTP/1.1 requests in an easy manner.
- **Bs4** (Beautifulsoup4) is a library provided by Richardson [35] that allows scraping of online information from HTML and XML files.
- **nltk.corpus**, a package consisting of modules that provide functions needed to read corpus files of different formats [21]. Such files contain data, including trained models. In the case of this work, it is used to identify stop words.
- **nltk.cluster.util** is a module used to cluster tokens [36], e.g., resulting from string tokenization.
- **nltk.tokenize**, a package that allows the division of strings into lists of substrings. It is used to find words and punctuation within a string [21].
- **datetime** is a module which provides classes for the manipulation of dates and times [37].
- **xml.dom**, a cross-language API from the World Wide Web Consortium that provides access and modification of XML documents [38].
- **pathlib** is a module that offers classes which represent filesystem paths with semantics that are appropriate for different operating systems [39].
- **xml.etree.ElementTree**, a module that implements an efficient API, allowing parsing and creation of XML data [40].
- **NumPy** as provided by the NumPy Developers [41] is a basic package for scientific computation in Python.
- **SciPy** [42] is used to perform scientific calculations with Python, it is recommended to install both NumPy and SciPy. SciPy is a collection of mathematical algorithms based on the NumPy extension of Python.
- **NetworkX** is a package that allows software written in Python to create, manipulate and study structure, dynamics, and functions of complex networks.

Appendix B. Code Script

In our prototype, URLs matching the user's inputs are first found with Google search before their content is scraped with BeautifulSoup4 and then summarized extractive using TextRank. The following steps are used in the code script:

1. Within the code script it is first defined how sentences are compared by using cosine similarity, which is a typical measure in text mining based on the frequency of word occurrences in two documents. This approach has the disadvantage that with an increasing query size, the execution of the program will take longer.
2. The method for creating the summary is defined. The tokenization, using the nltk library was added by the authors and the clean-up logic was improved.

3. We define how xml documents are created for the results. The results shall be readable using any text editor or Firefox.
4. For scraping, Google searches are performed. The authors modified the relevant piece of code by making the software ignorant of basic library calls. The program in question also pretends the search to originate from a Google Chrome browser on a Windows machine.
    a. The content from the resulting URLs is retrieved. URLs where content cannot be retrieved (e.g., because of using the noindex meta tag) are skipped automatically by the program.
    b. HTML String-Syntax is analyzed (parsed) with BeautifulSoup4.
    c. All paragraph elements are found and combined into a text. This excludes Headers.
    d. A summary of the text is generated using TextRank, an NLP module for extractive text summarization.
    e. Summaries of contents from the scrapable URLs are written into files.
5. The following user interaction in the form of user input is required to configure the program execution:
    a. search keywords
    b. top level domain, default = com
    c. number of results, default = 5

References

1. Wang, D.; Liang, Y.; Xu, D.; Feng, X.; Guan, R. A content-based recommender system for computer science publications. *Knowl.-Based Syst.* **2018**, *157*, 1–9. [CrossRef]
2. Lee, J.; Lee, K.; Kim, J.G. Personalized Academic Research Paper Recommendation System. 2013. Available online: http://arxiv.org/abs/1304.5457 (accessed on 25 January 2022).
3. Rajiv, S.; Navaneethan, C. Keyword weight optimization using gradient strategies in event focused web crawling. *Pattern Recognit. Lett.* **2021**, *142*, 3–10. [CrossRef]
4. Vural, A.G.; Cambazoglu, B.B.; Karagoz, P. Sentiment-focused Webcrawling. *ACM Trans. Web* **2014**, *8*, 1–21. [CrossRef]
5. Bifulco, I.; Cirillo, S.; Esposito, C.; Guadagni, R.; Polese, G. An intelligent system for focused crawling from Big Data sources. *Expert Syst. Appl.* **2021**, *184*, 115560. [CrossRef]
6. Axiotis, E.; Kontogiannis, A.; Kalpoutzakis, E.; Giannakopoulos, G. A Personalized Machine-Learning-Enabled Method for Efficient Research in Ethnopharmacology. The Case of the Southern Balkans and the Coastal Zone of Asia Minor. *Appl. Sci.* **2021**, *11*, 5826. [CrossRef]
7. Alarte, J.; Silva, J. Page-level main content extraction from heterogeneous webpages. *ACM Trans. Knowl. Discov. Data* **2021**, *15*, 1–105. [CrossRef]
8. Ferrara, E.; De Meo, P.; Fiumara, G.; Baumgartner, R. Web data extraction, applications and techniques: A survey. *Knowl.-Based Syst.* **2014**, *70*, 301–323. [CrossRef]
9. Baumgartner, R.; Gatterbauer, W.; Gottlob, G. Web Data Extraction System. In *Encyclopedia of Database Systems*, 2nd ed.; Springer: Cham, Switzerland, 2018.
10. Angulo, C.; Falomir, I.; Anguita, D.; Agell, N.; Cambria, E. Bridging cognitive models and recommender systems. *Cogn. Comput.* **2020**, *12*, 426–427. [CrossRef]
11. Dhelim, S.; Aung, N.; Bouras, M.A.; Ning, H.; Cambria, E. A survey on personality-aware recommendation systems. *Artif. Intell. Rev.* **2022**, *55*, 2409–2454. [CrossRef]
12. Karthikeyan, T.; Sekaran, K.; Ranjith, D.; Vinoth Kumar, V.; Balajee, J.M. Personalized content extraction and text classification using effective web scraping techniques. *Int. J. Web Portals* **2019**, *11*, 41–52. [CrossRef]
13. Carstensen, A.-K.; Bernhard, J. Design science research-a powerful tool for improving methods in engineering education research. *Eur. J. Eng. Educ.* **2018**, *44*, 85–102. [CrossRef]
14. Hevner, A.; Chatterjee, S. Design Research in Information Systems: Theory and Practice. In *Integrated Series in Information Systems*; Springer: Berlin/Heidelberg, Germany, 2010; Volume 22. [CrossRef]
15. Baskerville, R.; Wood-Harper, A. Diversity in information systems action research methods. *Eur. J. Inf. Syst.* **2017**, *7*, 90–107. [CrossRef]
16. Blum, F.H. Action Research—A Scientific Approach? *Philos. Sci.* **1955**, *22*, 1–7. [CrossRef]
17. Hevner, A.R.; March, S.T.; Park, J.; Ram, S. Design Science in Information Systems Research. *MIS Q.* **2004**, *28*, 75–105. [CrossRef]
18. IBM. What Is Natural Language Processing? Available online: https://www.ibm.com/cloud/learn/natural-language-processing (accessed on 20 October 2021).

19. Garbade, M.J. A Quick Introduction to Text Summarization in Machine Learning. Towards Data Science. 2018. Available online: https://towardsdatascience.com/a-quick-introduction-to-text-summarization-in-machine-learning-3d27ccf18a9f (accessed on 25 January 2022).
20. Zelle, J.M. *Python Programming: An Introduction to Computer Science*, 3rd ed.; Jones, B., Ed.; Franklin, Beedle & Associates Inc.: Portland, OR, USA, 2016; Available online: https://books.google.ch/books?id=aJQILlLxRmAC&printsec=frontcover&cd=1&source=gbs_ViewAPI&redir_esc=y#v=onepage&q&f=false (accessed on 25 January 2022).
21. Bird, S.; Loper, E.; Klein, E. *Natural Language Processing with Python*; O'Reilly Media Inc.: Sebastopol, CA, USA, 2009; Available online: https://www.nltk.org/ (accessed on 25 January 2022).
22. Ignatow, G.; Mihalcea, R. An Introduction to Text Mining: Research Design, Data Collection, and Analysis. In *An Introduction to Text Mining: Research Design, Data Collection, and Analysis*; SAGE Publications, Inc.: Thousand Oaks, CA, USA, 2018. [CrossRef]
23. Dubey, J.; Singh, D. A survey on web crawler. *Int. J. Electr. Comput. Eng. Syst.* **2013**, *1*, 1–5.
24. Dilmegani, C. What Is Web Crawling? How It Works & Examples. AI Multiple. 2020. Available online: https://research.aimultiple.com/web-crawler/ (accessed on 25 January 2022).
25. Castellano, M.; Fiorino, F.; Arcieri, F.; Summo, V.; de Grecis, G.B. A Web Mining process for e-Knowledge services. *WIT Trans. Inf. Commun. Technol.* **2006**, *37*, 12.
26. vanden Broucke, S.; Baesens, B. Practical Web Scraping for Data Science. In *Practical Web Scraping for Data Science*; Apress: New York, NY, USA, 2018. [CrossRef]
27. De, S.; Sirisuriya, S. A Comparative Study on Web Scraping. 2015. Available online: http://ir.kdu.ac.lk/bitstream/handle/345/1051/com-059.pdf?sequence=1&isAllowed=y (accessed on 25 January 2022).
28. Python Software Foundation. The Python Language Reference—Python 3.9.9 Documentation. Available online: https://docs.python.org/3.9/reference/ (accessed on 13 January 2022).
29. Anaconda Inc. Anaconda Individual Edition—Anaconda Documentation. 2021. Available online: https://docs.anaconda.com/anaconda/ (accessed on 13 January 2022).
30. Microsoft. Documentation for Visual Studio Code. Available online: https://code.visualstudio.com/docs (accessed on 13 January 2022).
31. Rehurek, R. Gensim. 2010. Available online: https://pypi.org/project/gensim/ (accessed on 25 January 2022).
32. Vikramaditya, N. Googlesearch-Python PyPI. 2021. Available online: https://pypi.org/project/googlesearch-python/ (accessed on 25 January 2022).
33. Klein, E. NLTK: Nltk.sem.evaluate. Available online: https://www.nltk.org/_modules/nltk/sem/evaluate.html (accessed on 25 January 2022).
34. Reitz, K. Requests PyPI. Available online: https://pypi.org/project/requests/ (accessed on 25 January 2022).
35. Richardson, L. Beautifulsoup4 PyPI. Available online: https://pypi.org/project/beautifulsoup4/ (accessed on 25 January 2022).
36. Cohn, T. NLTK: Nltk.cluster.util. Available online: https://www.nltk.org/_modules/nltk/cluster/util.html (accessed on 25 January 2022).
37. The Python Software Foundation. Datetime—Basic Date and Time Types—Python 3.10.2 Documentation. Available online: https://docs.python.org/3/library/datetime.html (accessed on 25 January 2022).
38. The Python Software Foundation. xml.dom—The Document Object Model API—Python 3.10.2 Documentation. Available online: https://docs.python.org/3/library/xml.dom.html (accessed on 25 January 2022).
39. The Python Software Foundation. Pathlib—Object-Oriented Filesystem Paths—Python 3.10.2 Documentation. Available online: https://docs.python.org/3/library/pathlib.html (accessed on 25 January 2022).
40. The Python Software Foundation. xml.etree.ElementTree—The ElementTree XML API—Python 3.10.2 Documentation. Available online: https://docs.python.org/3/library/xml.etree.elementtree.html (accessed on 25 January 2022).
41. NumPy Developers. NumPy Documentation—NumPy v1.22 Manual. Available online: https://numpy.org/doc/stable/ (accessed on 25 January 2022).
42. SciPy. SciPy 1.7.3. 2019. Available online: https://scipy.org/ (accessed on 25 January 2022).