15 Low-code web scraping and text analysis with Octoparse and KNIME: An example from the CICuW project

Daniel Ihrmark
Linnæus University

Hanna Carlsson
Linnæus University

Fredrik Hanell
Linnæus University

Low-code tools play an important role in making data analysis and visualization accessible to researchers and students with limited experience, or interest, in programming. While low-code tools do introduce closed-box issues, they can still be considered important stepping stones toward computational approaches. This chapter draws on two such tools, Octoparse and KNIME (Konstanz Information Miner), to present a workflow from data collection from online sources, through text pre-processing, toward text classification in the context of the ongoing project Cultural Institutions and the Culture War (CICuW) that investigates the democratic implications of the pervasiveness of far-right digital discourse.

This chapter will introduce web scraping, topic modeling, and sentiment analysis in an accessible way, while also showcasing state-of-the-art approaches to the analysis components through the use of BERT (Bidirectional Encoder Representations from Transformers) models and zero-shot classification. The chapter will take a critical perspective on the described methods by discussing how they contribute to creating methodological closed-boxes and how quantitative techniques can be fruitfully combined with qualitative approaches.

1 *Introduction*

In Sweden, the growing influence of the far-right has turned cultural institutions into political symbols in an emerging "culture war" (Harding 2022). Carlsson et al. (2022) show how digital forums and social media play a significant part in orchestrating the ideologically laden conflicts and

HUM INFRA Daniel Ihrmark, Hanna Carlsson & Fredrik Hanell. 2025. Low-code web scraping and text analysis with Octoparse and KNIME: An example from the CICuW project. In Gerlof Bouma, Dana Dannélls, Dimitrios Kokkinakis & Elena Volodina (eds.), *Huminfra handbook: Empowering digital and experimental humanities* (NEALT Proceedings Series 59), 505–540. University of Tartu Library. DOI: 10.58009/aere-perennius0184

confrontations that public libraries are currently facing. Digitally mediated threats from the far-right may obstruct the statutory mission of the institution to promote democracy, but knowledge about how such threats develop and unfold, as well as the relation between online interactions and offline events (Scrivens et al. 2020), is lacking. These insights form the starting point for the ongoing research project Cultural Institutions and the Culture War (CICuW) that focuses the democratic implications of the pervasiveness of far-right digital discourse through an investigation of the digital environments where supporters of far-right ideas interact, share thoughts, construct collective identities, and coordinate actions toward libraries.

This chapter details the methods involved in conducting a pilot project of such an investigation, using online interactions as a sample explored through sentiment analysis and topic modeling. The sample in the original project was collected via web scraping, which is also described in the chapter. However, as the intended audience of this handbook is wide, both in terms of field of study and in languages spoken, the use case provided in the chapter will not make use of the Swedish language data explored in the pilot study and instead focus on creating a more general English language dataset.

The chapter begins with a short discussion regarding the use cases for the methods being introduced and some of the ethical considerations at play when working with online data collection, such as web scraping. The authors intend that readers should be able to follow along with their own project while reading the chapter, regardless of whether they make use of the same data sources as we do or decide to follow along with a data source of their own. However, the introductory sections must be read, understood, and considered before the work begins. Sections 2 to 5 make up what could be considered the pre-reading before the active participation begins in Section 6.

2 *Research problem and relevance to Digital Humanities*

Text analysis is by no means a new addition to the field of Digital Humanities, and could be considered the starting point for the field through the work of Roberto Busa and IBM during the early days (Jones 2016). The techniques included in text analysis are also often found at the intersection Digital Humanities share with Corpus Linguistics, although the aims of the two fields can often be seen as quite different. While linguists, as the name would suggest, are often concerned with language features and the development of language over time, digital humanists who take an interest in text analysis are instead more often concerned with the content of the language or what is being expressed. While these two groups often overlap, it is important to

note that the techniques discussed in this chapter fall into the latter category, and are aimed at producing content overviews of a collection of texts for further exploration.

This aim brings us to a second important thing to note about the contents of the chapter, namely that the project upon which it is based as well as the project introduced in the walkthrough components would fall under what is considered mixed-methods research. Mixed-methods research refers to an approach that mixes quantitative and qualitative research in order to arrive at its conclusions. For Digital Humanities projects, this often becomes the mode of combining perspectives from more traditional humanities with results derived from computational methodologies. For instance, in the CICuW project, quantitative overview methods such as sentiment analysis and topic modeling are used to create narrower focal points for qualitative methods, which in greater detail explore the discourse surrounding the selected topics and expressions of sentiment. While this chapter emphasizes the computational methods found at the first stages of exploration, the actual results of the pilot are intended to be formulated based on the latter qualitative component. As a whole, CICuW aims to offer a timely investigation into the democratic consequences of the pervasiveness of far-right discourse, focusing online information activities, on-site experiences, and their interconnectedness. In this way, digital methods commonly used withing the Digital Humanties are used in novel ways and combined with qualitative procedures to explore the nature of far-right digital discourse targeting cultural institutions and the relations between online far-right discourse and on-site hostilities.

In addition to text analysis, this chapter also includes web scraping as a mode of data collection. Briefly described, web scraping is a method for collecting materials from webpages and can be used for the collection of text, images, or any other component of a website. Many different technologies are available for web scraping, but they often rely on the structural element of the webpage in order to find the correct data to collect. This means that in order for web scraping to be efficiently applied, some basic know-how regarding webpages and web technologies is needed.

Furthermore, as we start to interact with the world wide web, privacy concerns and ethics become increasingly important. The online world provides us access to immense amounts of information of different types and a lot of that information is provided by the users of the web pages we are collecting information from. While data collection involving humans would normally involve informed consent and ethical approval, web scraping of online data is a bit more tricky. In this chapter, we will discuss the ethics of web scraping and online data collection based on the Association of Internet

Researchers (AoIR) ethics framework (franzke et al. 2020), but it is important to also consider the national context and institutional policy that might be relevant to your project.

3 *Research/use scenarios*

3.1 *Web scraping*

As mentioned in the introduction, web scraping is all about collecting information from the world wide web. This use case is often accomplished by interacting with a web browser through automation and then collecting the relevant components of whatever website or set of websites that one might be interested in. Automating the web browser interaction is the key to the usefulness of web scraping as a technique, as it allows for collection of large amounts of data quite quickly. While this chapter is focused on the use of web scraping for collecting text data, webpages also contain many other modalities which could be collected using a similar approach. In order to successfully perform data collection through the use of web scraping in a research context, the main components to be conscious of would be (1) the design of the target websites, (2) the type of automation involved, and (3) the ethical considerations at play in the given context. These will be briefly explained in regards to their relationship to web scraping in the following sections.

3.1.1 *Web sites*

At their core, web pages could be seen as explicitly structured documents containing different kinds of information and materials. The primary structure of a web page consists of HyperText Markup Language (HTML), which organizes all of the page's different types of content and provides the skeleton structure of where the pieces on when displayed in a browser. So, while we might see a visual representation of a web page projected for us in the browser, the actual web page consists of structured information. To have a quick look at how this might look, one can use the "Inspector" feature included in most web browsers and navigate to the "Elements" tab. An example of the Huminfra main landing page viewed through the "Inspector" in a Google Chrome browser is seen in Figure 1.

This structured information is key for the browser to be able to correctly display a page according to the designers instruction. Furthermore, it is this very structure that we hijack in order to successfully navigate the web page to collect relevant data when web scraping.

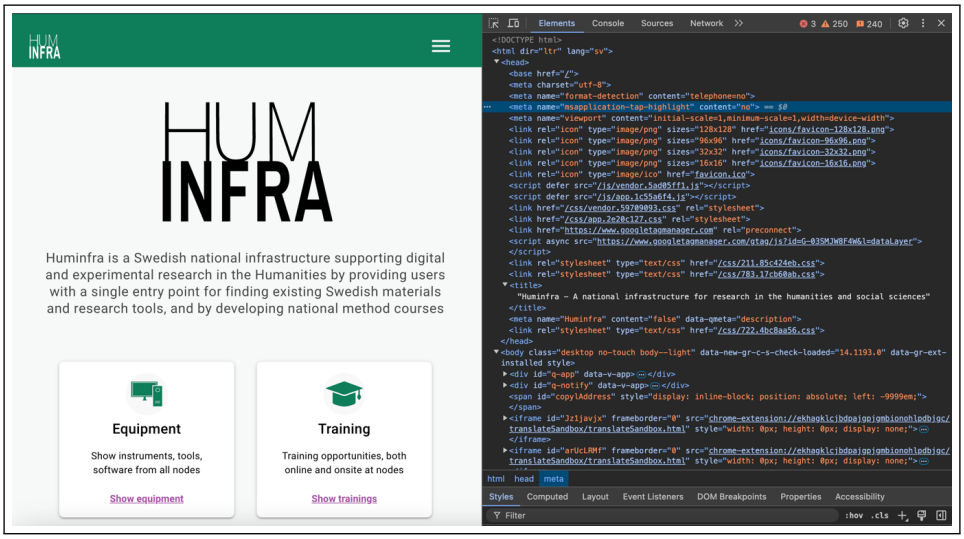

Figure 1: The Huminfra landing page Elements tab seen through the Inspector view of a Google Chrome browser.

The HTML structure that makes up the web site consists of elements, each enclosed by tags that describe whatever content is found inside. Returning to the example in Figure 1, we can see that the tag "`<title>`" encloses the text "Huminfra – a national infrastructure for research in the humanities and social sciences", which makes up the title element of the web site being displayed, for instance, in the browser tab where the page is open. We also see quite a few `<div>` tags, which are used to group elements into divisions. The different tags provide the hierarchy of the structured information, and make up what is known as the Document Object Model (DOM).

In order for our web scraper to be able to select the appropriate component of the web site to collect, we will need to specify a specific part of the DOM either by use of tags or named divisions. However, not all web sites are structured the same and the more diverse our intended target sites are the more complicated this selection process is. For the example in this chapter, we are going to collect information from one single web site. This means that we can assume that the structure will be the same for each web page will be largely the same in terms of its structure and element naming scheme.

3.1.2 *Automation*

Now that we have a better understanding of the way that web sites are structured, we can move on to discussing how we avoid collecting material manually from one place at the time. This is where automation shines, as it enables the collection of large amounts of data with minimal human intervention (although some is always needed). In addition to allowing us to avoid the manual portion of collecting the materials, automation also allows research projects to perform continous data collection or to set up a schedule of web scraping at intervalls. This allows research projects to scale up and to start engaging with the large amounts of materials often preffered for digital methods.

In the typical web scraping scenario, automation is applied through the use of a programming script drawing on the diverse multitude of resources available through programming languages such as Python or through the use of specialized software which is already configured to perform web scraping tasks. This chapter will make use of one such software tool, Octoparse (2024), which will be introduced in a later section. The automation needs to be able to perform the main tasks of identifying and collecting the appropriate data, and many solutions will also include the functionality to clean the data to a certain extent.

However, automation will often have to extend further than just these basic tasks. Due to the interactive nature of many web pages, web scraping tools will also often need to be able to interact with a site the way that a user would, using the browser interface to navigate the site, open up sub-pages, and occasionally entering information in order to search for materials or perform actions requiring a user to be logged in. This is especially true when interacting with web pages containing JavaScript, which creates dynamic web content which might not be loaded onto the page until certain interactions are performed. In programming, this is accomplished through the use of resources such as Selenium (Selenium 2024), but for software tools it is something that will need to be taken into account when selecting the right tool for the job.

While automation is highly beneficial and provides some of the main benefits of picking up web scraping as a skill, it also comes with a heightened level of responsibilty when compared to manual data collection. Web sites will often implement security measures such as login prompts or CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) in order to counter automated attacks or limit the amount of interactions made with their content at a given time. While performing automated interactions with a web site, it is our responsibility to ensure that we do so

without creating an issue for the owner of the web site or the host. This includes limiting the number of interactions done in line with what would be considered reasonable, making sure that we are not clogging up the access to the website while collecting our materials, and making sure that we are not collecting things that we should not collect. This is where the ethical considerations come into play.

3.1.3 *Ethical Considerations*

As web scrapers rely heavily on automation and since their purpose is essentially the autonomous collection of data and performance of tasks, they are sometimes discussed as robots. However, regardless of how automated and autonomous the collection procedure is, the ethical considerations must still be thouroughly engaged with by the researcher. Depending on the size and topic of the project the ethics involved can vary in complexity, but it is an area in which one should always prefer to have done too much rather than too little.

The Internet Research Ethics (IRE) is a document containing ethics guidelines for research involving materials and data taken from the web, and is continually being worked on by the Association of Internet Researchers. At the time of writing this chapter the IRE is in its third version, but readers after the year of publication should consult the current version for the most up-to-date information. While this chapter will rely on the IRE 3.0 (franzke et al. 2020) as its main document of ethical consideration, it is very important that any researcher engaging in web scraping also consult policies or regulations in place at their university or otherwise relevant to their immediate context.

Returning to the possible harm caused by web scraping, the considerations are often due to the nature or information available online. As the IRE 3.0 guidelines highlight, internet research must balance the benefits of scraping large datasets with the potential for harm to individuals whose data is being collected. Even in cases where data appears publicly accessible, such as on social media or forums, researchers should remain critical about the expectations of the users whose data they scrape. Users may not always be aware that their data is being collected for research, leading to a potential violation of their privacy. In addition, considerations must also be made regarding the potential use of the data being collected and the possibility of linking said data to the account from which it originated, and further on from there to an actual person. This is especially important when collecting potentially sensitive data from web pages on which interactions amongst vulnerable groups take place, such as Reddit sub-forums connected to sexual

orientation or religion. These examples would also be representative of the types of information that might be unethical collect according to local or national policy and regulation in certain cases.

In addition to the general guidelines of ethical internet research, web sites can also supply information about which types of information are allowed to be collected. These limitations are expressed in a robots.txt file, which should be available by simply adding "/robots.txt" after the main address of the site. The robots.txt is essentially a web standard that allows website administrators to communicate which parts of a site should not be accessed by automated robots. Respecting the instructions in a robots.txt file is a baseline ethical obligation for researchers as it represents the website owner's consent—or lack thereof—for their content to be collected. Ignoring robots.txt can lead to potential legal and ethical violations through accessing data that the site owner has explicitly marked as not to be collected. Researchers could see this as a form of informed consent in the digital realm, similar to the expectations outlined in the IRE 3.0 guidelines. However, it is very important to note that the ethical considerations that go into collecting online data for research go beyond the robots.txt, as research projects are expected to engage in a more comprehensive view of the ethics involved in their methods.

The ethical considerations do not end once the data has been collected, but must also extend to the way that the data is saved and stored. Depending on the sensitivity of the data, ethical policies and guidelines might dictate that certain precautions be taken in terms of anonymizaton and storage in order for confidentiality to be maintained. For these reasons projects collecting data through web scraping might also need to include use of a data mangement plan as a part of their design, especially so when submitted for formal ethics review. Norms underpinning principles for ethical research practices include "respect for persons, beneficence, and justice" (franzke et al. 2020: 4). These norms are important both when doing quantitative and qualitative studies online. In the context of qualitative studies, additional emphasis is often placed on reflexivity, the situatedness of human practices, and how to balance risks for participants against the potential value of the planned research (cf. Kozinets 2020). Such insights can also enrich method-driven studies often associated with the Digital Humanities (Hanell & Severson 2023).

3.2 *Text analysis*

Once we have completed web scraping and have a collection of texts to work with, we can start applying text analysis to explore the new dataset. As mentioned previously, the text analysis techniques discussed in this chapter

are aimed at producing content overviews which can serve as a starting point for further work. The end result of applying text analysis, in this specific use case, is therefore not an analysis of the materials but rather serve to make it possible to identify components of the materials for further qualitative assessment. Due to the overarching research design of the project which the chapter is based on, the output of the methods used are exploratory visualizations that can be used to explore the data.

Text analysis is an umbrella term which encompasses a myriad of different techniques, often with the purposes of identifying patterns in textual data, creating overview of different features in the text, or to annotate texts according to whatever classification might be relevant for a given project. In this chapter, we focus on topic modelling, which identifies patterns of topics in a text collection, and sentiment analysis, which annotates texts based on approximations of expressed sentiment.

3.2.1 *Zero-shot sentiment analysis with BERT*

Sentiment analysis, as mentioned in the previous paragraph, is all about figuring out the sentiment expressed in a text. Sentiment in this context is understood along the lines of emotional tone, and is often categorized as being positive (I love this!), negative (I hate this...), or neutral. Do note the missing example for the last category, as it often ends up being the bin category for texts in which neither positive or negative sentiments are found. While this chapter is going to focus on zero-shot sentiment analysis, as used in the CICuW project, we are going to start by discussing two other approaches to the concept, lexicon-based and supervised machine learning-based, that will serve to highlight the specific method choice made for the project.

Lexicon-based sentiment analysis, as the name suggests, relies on a lexicon comprised of sentiment-carrying terms which are accumulatively identified in the text. The number of positive terms and negative terms are added up, and the total is considered the sentiment of the text. The technique does not only count the words, of course, but instead relies on sentiment weights being attached to each term in the lexicon, identifying them as more or less negative or positive. In addition, more advanced lexicon-based methods such as VADER (Hutto & Gilbert 2014) also take the context of the term into consideraton and accounts for expressions of intensity as well as negation. Considerations when applying a lexicon-based approach should include how well the lexicon fits with the context to which it is applied. For instance, VADER (Valence Aware Dictionary and sEntiment Reasoner) was formulated specifically for microblog-like contexts, such as social media.

Supervised machine learning-based sentiment analysis may sound intimidating, but it is actually quite easy to apply once the pre-requisite components are assembled. The approach relies on using a machine-learning model to identify sentiments based on pre-labelled examples, the so-called training data, and then apply the same labels to previously unseen texts. Unlike lexicon-based methods, which rely on pre-defined word lists, machine learning approaches can capture more nuanced relationships in the training data by using algorithms such as support vector machines, logistic regression, or neural networks. The key consideration when using supervised machine learning for sentiment analysis is the quality and size of the training data, as the model's performance relies on having a representative dataset to learn from. However, representative is not a general quality in this case, and must be understood contextually. Training a model on emails and the applying it to YouTube comments is unlikely to produce viable results.

Zero-shot sentiment analysis makes use of large models, such as BERT (Bidirectional Encoder Representations from Transformers, Devlin et al. 2019), to identify expressed sentiment without the use of a specific lexicon or a training dataset. This relies in part on the massive amounts of data that have already been used to train the model, which allows it to infer the sentiment from an unseen text based on the labels provided even when explicit examples have not been provided. The model is prompted with a set of labels, for instance Positive, Negative or Neutral, and then evaluates the new text according to what the model already has available for, or "knows" about, each of the provided labels.

When compared to traditional supervised models, zero-shot sentiment analysis stands out by removing the dependence on labeled datasets, leading to a higher potential scalability and adaptability to new contexts. This can be an especially important feature of the technique when one is conducting a project where the available data is limited, or where there are no resources for manually labelling a large enough quantity of training data. However, the performance of zero-shot models may not always match that of supervised methods fine-tuned for a specific task if large amounts of labeled data are available, as supervised models can optimize their predictions based on that dataset.

3.2.2 *Topic modelling with LDA*

While there are many different techniques available for applying topic modelling, they all have in common that the end goal is to identify latent topics or themes. What is meant by latent here is that the topics or themes do not have to be explicitly defined before the technique is applied, but are rather

created by the method by applying some kind of conceptualization of what a topic should be to the data. Oftentime, we consider a topic to be made up of content words that co-occur with one another frequently within the collected texts.

One of the more popular methods relying on this rationale is Latent Dirichlet Allocation (LDA), introduced by Blei et al. (2003). The method represents a perspective on topics in texts which assumes that every text is in fact made up of multiple topics which are distributed across the texts in a collection. Since topics are created dynamically based on the content of the texts in the collection and since LDA produces the distribution of topics amongst the texts, the results are very useful for identifying texts that are mainly on one topic but might include other topics to a lesser extent. This is especially useful when applying topic modelling as a mode of exploration, as we are doing here.

In order to apply LDA we must first specify certain variables, such as the number of topics to be extracted and the number of terms to be included within each topic. Here we have two separate options: Qualitative assessment of the results or quantitative optimization. The project this chapter is based on decided on qualtitative assessment due to the texts in the collection being short in length and due to the specific interests of the the later qualitative analysis. In practice, this meant iterating over the collection with different settings and deciding on the number of topics and terms that ended up producing the most relevant results in terms of the content described in each topic.

For quantitative optimization of LDA, the "elbow method" is often applied. The elbow method involves running the LDA model with different numbers of topics and plotting a metric representing the quality of the topic, such as a perplexity or coherence score, against the number of topics. As the number of topics increases, the model improves, but after a certain point, the improvement slows down, forming the eponymous "elbow" in the plot. This would mean that the number of topics used just before the bend is the most optimal.

However, before we start topic modelling our text collection we must spend some time on pre-processing the texts. Since the LDA method relies on frequent co-occurrence of terms, we want to make sure that the terms included in each topic are relevant ones, previously discussed as "content words". From an extremely reductive perspective, we can split language into two categories: content words which convey content, and function words which perform a syntactical function within the given grammatical system. Examples of the latter category would be words like conjunctions and articles. The pre-processing involved in successfully applying co-occurrence-based

topic modelling will often involve stopword removal, which removes function words from the text. This is based on a stopword list, which contains all of the words to be removed. It is important to be aware of what is being removed from the text, as different stopword lists contain different words and vary greatly in length.

In addition to stopword removal, pre-processing can also include techniques to condense the text. The purpose here is to remove different variants of the same word resulting in more correct frequencies for the concepts represented by the words and less noise in the results. This could for instance be done through lemmatization, which replaces every variant of a word with its dictionary form. Finally, punctuation is often also removed. Here it is important to be aware of all the changes made, as the pre-processed text to which the method is finally applied is quite different in its form than the original text first collected. It is also important to keep a keen eye on whether a method is being fed the pre-processed text or the original, as it can greatly influence the outcome.

3.3 *Exploratory information visualization*

While seen here from a narrow view in its role as the output of the text analysis components of the project, it is important to be aware of the fact that information visualization is a field of research in its own right. The functionality of different types of information visualization of course depends on use case and purpose, but for our approach we are mainly interested in interactive visualizations that allow us to overview the large dataset and find items that are relevant for our manual analysis. Shneiderman (1996) is often quoted as describing the design rationale for such interactive visualizations as: "Overview first, zoom and filter, and then details on-demand", which provides an idea of both the needed functionality as well as guidelines for intended usage. However, we also have to consider the types of information we want to visualize.

In the case of the project on which this chapter is based, we are mainly interested in visualizing numerical values derived from sentiment analysis (negative to positive) and topic modelling (distribution of topics across different items). However, the latter must also include a representation of the terms included in each topic for visualization legibility. The individual texts make up the relevant granularity of our visualizations, as the main purpose of the visualizations produced will be to identify relevant texts within the sample. Considering this with Shneiderman's mantra in mind, we will need exploratory visualizations that allow for an overview of the results from the sentiment analysis and topic modelling, zoom and filtering

including the selection of individual texts, and details that will allow for the selection of individual texts for further study.

4 *Resource / tool description*

4.1 *Low-Code and No-Code Tools*

The broad group of tools referred to as low-code or no-code should in the context discussed here be understood as digital environments which allow for the user to create workflows and applications with, as the terms suggest, little or no actual programming needing to take place. The environments often utilize a drag-and-drop interface which allows the user to combine pre-built components and functionalities in a visual workflow. These tools are intended to maintain a balance between transparency and ease of use. While not involving programming, the use of low-code and no-code tools still involves a level of computational thinking and problem decomposition which makes them a good point of entry to computational text analysis methods (Tyrkkö & Ihrmark 2024).

Computational thinking, as defined by (Wing 2006), is a problem-solving process that involves various cognitive skills, such as abstraction, algorithmic thinking, decomposition, and pattern recognition. Essentially, it is intended as a way of tackling a problem by breaking it down into its components and then designing a workflow that answers to the identified components. This is also the approach needed for work with low-code and no-code tools, as the conceptual and abstract work of designing the process remains just as important as it would when programming, just with different building blocks.

These types of tools also introduce concerns regarding the use of closed-box approaches within research. The term closed-box refers here to the way that fairly complicated methods can now be employed with relative ease, without requiring the researcher to necessarily comprehend exactly how the methods work or arrive at their results. While low-code and no-code tools allow us to experiment with and apply new and interesting methods, the responsibility of understanding what is happening is still on the researchers. This also includes understanding the biases, assumptions, and limitations of every method as applied in whatever context is being engaged with.

In this chapter, the context for application is both data collection and analysis. For data collection, considerations need to be made regarding what is collected and how it is stored, as discussed in 3.1.3. For the text analysis workflow, the focus is more on understanding the components of the

workflow and the rationale of the methods in order to produce reliable and comprehensible results. The first tool introduced is Octoparse, which will be used for web scraping. This is followed by KNIME (Konstanz Information Miner, Berthold et al. 2009), which will be used for the text analysis.

4.2 *Octoparse 8*

Octoparse 8 is intended as a no-code tool that allows for automated data collection from online sources. In other words, it is a web scraper with a point-and-click interface that enables data collection without a need for programming. The tool allows for collection from both static web pages and dynamic pages with JavaScript elements. It also offers functionality for storing login information and automating access that might be behind a password barrier, as well as support for browser interactions that can allow for navigating the site or closing pop-ups. The tool shows the provided URL in a browser window, and allows the user to either select elements for collection manually or through the auto-detect feature.

The auto-detect feature, which maps the page elements and generates the parsing rules needed to identify them within a group of pages with similar structures, is one of the main functionalities that makes Octoparse a great entry point into web scraping. This skips some of the more engaged steps in conducting manual scraping, such as reading the page's HTML and figuring out the connection between its tags and the relevant content. However, the feature is not flawless and some fine-tuning is to be expected in most cases, meaning that some understanding of the web page structure is still needed. Figure 2 showcases the tool browser after having applied the auto-detect feature to the IMDb (Internet Movie Database) page for the 1999 classic "Office Space", resulting in a dataset consisting of the actors in the film alongside their roles. The resulting dataset is showcased at the bottom of the tool window.

Like many no-code tools, Octoparse works based on a workflow. The workflow dictates the steps of navigating the site, entering information, locating relevant information, and collecting it. The workflow also allows for the creation of loops as a part of the process, for instance enabling collection to be possible iteratively based on a list of URLs on the site. In Figure 3, the workflow for a scraping task based on a list of URLs is shown. For common tasks, such as data collection from social media, the tool provides task templates.

Whether one wants to compile a list of specific URLs for scraping or collect data based on a list of links already available on a website depends on goals of the project. For the CICuW project use-case, the target website's

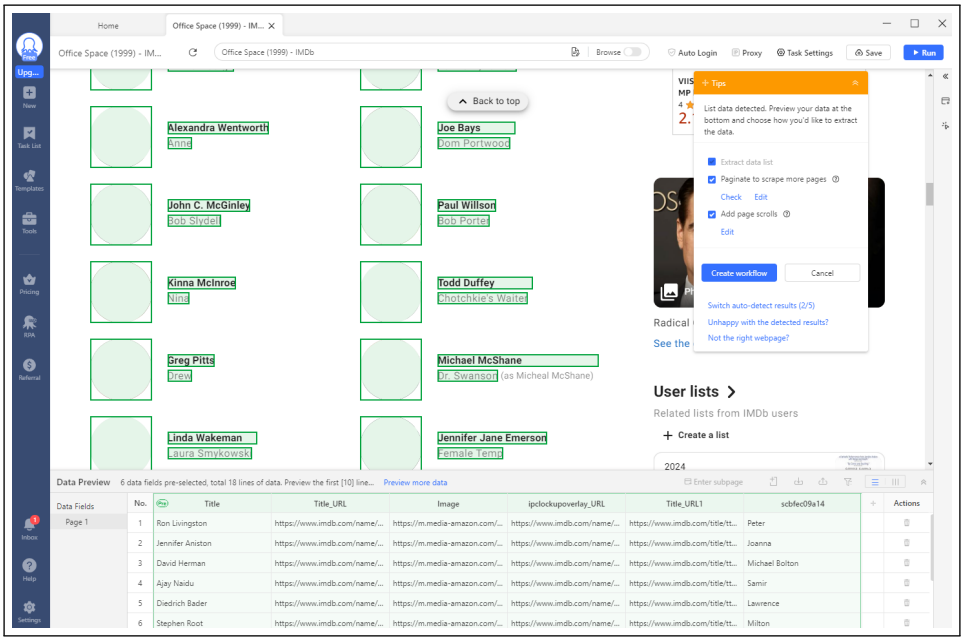

Figure 2: Octoparse interface showcasing extracted elements from IMDb page for "Office Space" (1999).

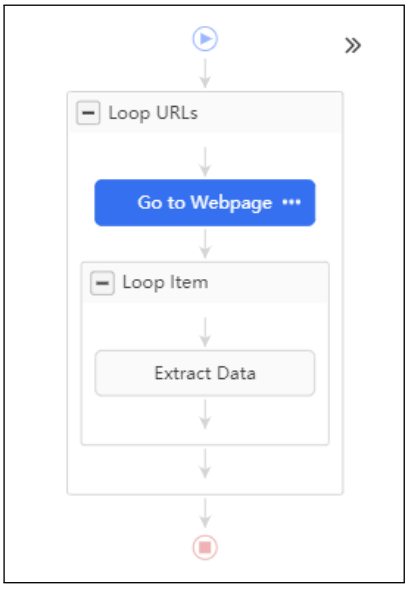

Figure 3: Octoparse workflow.

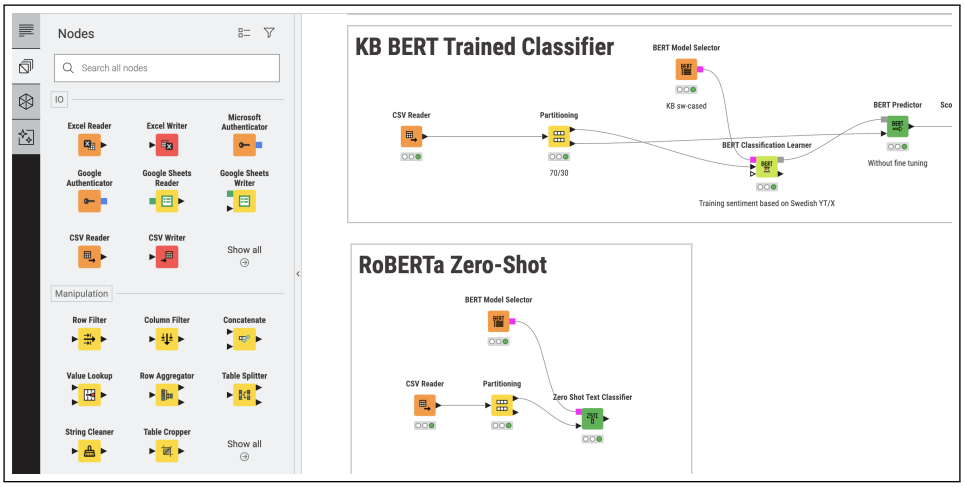

Figure 4: Workflow with node sidebar shown on the left.

search function was used to find the relevant materials, which created a list of links on the page to work based on. However, for other collection targets, one might prefer to manually curate a list of URLs for greater control over what is included in the resulting dataset.

4.3 *KNIME*

KNIME is a powerful, open-source, low-code tool that offers many different types of analysis. In addition, KNIME is also capable of producing interactive visualizations as a part of the workflow, meaning that it is a good fit for cases where the visualizations are used to filter data for further work. It is worth noting that KNIME is defined as a low-code tool, as it offers the opportunity to include programming scripts in the workflow to further extend its functionalities. However, no programming will be necessary for the uses showcased in this chapter.

The core of KNIME is its visual workflow editor, which is used to string prebuilt functions together for process being conducted. The functions are represented by nodes, which visually indicate the placement of the function in the workflow. The workflow is visualized on the canvas component of the tool's interface, into which nodes are dragged and dropped from the sidebar. The order of the workflow is determined by how the nodes are connected. The example workflow for this chapter is shown in Figure 4. We recommend downloading the workflow for use in the walkthrough via the

KNIME Hub[1], as importing it to your local KNIME installation will also prompt the installation of the needed extensions.

In a KNIME workflow, the data flows along the workflow through the connections made between the nodes. Nodes operate on a "Traffic Light" signal system, where nodes with a red marker beneath them are in need of either an input or configuration, yellow markers indicate a node ready to execute, and green markers indicate nodes that have been successfully executed. The way we work with a KNIME workflow is first designing the process, then assembling the nodes, and finally running them in our decided order so that we can perform whatever task we have in mind with the input data.

When KNIME is first installed, it includes some common or general prebuilt functions. In order for the tool to be useful for specific tasks, such as text analysis, the available nodes must be expanded through the installation of extensions. Extensions are available via the KNIME Hub, and can be installed via the tool or via web browser. The KNIME hub hosts both official KNIME extensions and community extensions, which are created by the users of the tool. Some specialized extensions are not available via the hub, and need a more engaged process which includes manually specifying the correct source repository in order to be installed.

The extensions highlight both a main drawback and a main benefit of KNIME as a tool. The use of extensions allows for the possible use cases for KNIME to be continuously expanded according to the needs of the user community, but it also creates a situation where problems can arise from changes in one extension creating compatibility errors with other extensions that are developed separately but are often used in combination. Furthermore, the functions, processes and tasks afforded by the available nodes and extensions at any given time provide the limit of KNIME's usability, with uptake often lagging a bit behind cutting-edge tools and approaches possible via programming.

In summary, KNIME provides an entry point to computational methods with a significantly lower barrier when compared to programming languages such as Python, but it also lags behind the developments in methods available through such approaches. When used without appropriate consideration, low-code tools like KNIME can also exacerbate closed-box issues. However, the use of the canvas to explicitly connect nodes in order to create a workflow where the data can be inspected and interacted with at any point during the process allows for tasks to be engaged with in a way that supports computational thinking and can help in avoiding haphazard use of methods in cases where the user remains engaged with the different components involved.

1 https://hub.knime.com/s/qDeklEsCiNtekxfv

5 *Case study: Cultural Institutions and the Culture War*

As previously described, public libraries in Sweden have become a focal point in the growing cultural conflict connected to the rise of the far-right. Digital forums and social media play a significant role in fueling these ideologically driven confrontations, which may threaten the libraries' democratic mission. However, there is limited understanding of how these online threats evolve and their connection to real-world events, highlighting the need for further investigation into digital spaces where far-right supporters organize and coordinate actions against libraries. The CICuW project aims to investigate this connection between online and offline interactions between the far-right and the public libraries through analysis of online posts on far-right platforms and offline incidents at public libraries.

The project consists of three sub-projects, of which one is engaged with the types of methods discussed in the current chapter. However, it is important to note this as a component of the project, as it ties into the mixed-methods context in which it takes place. This essentially means that the project as a whole involves both qualitative and quantitative methods. The three interconnected sub-projects involved in the overarching CICuW project are focused on three different components of the research context: digital information activities in far-right spaces, on-site experiences at cultural institutions, and finally the interconnections between the two. The current chapter is based on a pilot study taking place within the first sub-project, dealing with online information activities.

The pilot study was conducted in order to explore the use case for different methods in a smaller sample of relevant materials. The data for the study was collected from an online news platform connected to the Swedish far-right. Articles about public libraries were collected using the site's internal search functionality, and both the articles themselves and the user comments on the individual articles were collected. The different kinds of data collected allowed for the analysis to be conducted on two levels: articles and comments.

The articles were analyzed for topic and sentiments in order to provide a starting point for the topic and sentiment of the exchanges on the platform, essentially acting as the tone-setter for the interaction. In addition, the articles could also be seen as providing both a spatial and temporal grounding for the exchanges in the comments. The comments, on the other hand, could be seen as representing the reaction to the article, and as indicating which topics or sentiments were picked up for further discussion. The number of comments is, of course, also indicative of the level of engagement with a given topic.

Once analyzed using topic modelling and sentiment analysis, the data was further engaged with through qualitative content analysis (Hsieh & Shannon 2005). In practice, this means that the results of the topic analysis was engaged with in order to understand what each of the resultant term collections could refer to in the context of the sample, and a close-reading of relevant pieces was then conducted in order to verify the interpretation. This qualitative assessment also provides the conclusions of the study, as the deeper understanding of the material needed to answer the research questions is produced. This highlights the exploratory use of the digital methods in the context of CICuW, as they serve to filter the data and produce smaller samples to be engaged with directly. While the data we will engage with in the walkthrough is quite different both in nature and language, the process and steps taken are the same.

6 *Example walkthrough using IMDb*

For our example walkthrough, we will collect the descriptions of the top 100 rated movies on the Internet Movie Database (IMDb). We will also collect some additional information, such as user ranking and year of release. However, before we start deciding on exactly what we want to collect, we must first have a look at the site's robots.txt (Figure 5) in order to see if any of the specific information we are interested in is disallowed.

The disallowed items on the page seem to have mainly to do with either dynamically generated content, such as "on this day" or searches, and user-specific content. We can therefore carry on with our intended collection as planned.

The basis for our dataset is going to be the list of movies provided via the "Top 100" link on the website's opening page. The link takes us to a list of films, sorted on user score. We will be using Octoparse to iterate through the list by essentially clicking the URLs in the order provided to us, and collect information from each of the pages. This way we do not have to compile a list of URLs ourselves, and can instead rely on the structure already in place on the site.

6.1 *Collecting the Data with Octoparse 8*

We start by opening Octoparse and clicking "Custom Task" in the hover menu that appears when we hover "New" in the menu on the left. This will prom,pt us to supply either a .csv file with URLs or to input the URL into the text box. Since we will be working with a single URL, we simply copy

```
# robots.txt for https://www.imdb.com properties
User-agent: *
Disallow: /OnThisDay
Disallow: /ads/
Disallow: /ap/
Disallow: /mymovies/
Disallow: /r/
Disallow: /register
Disallow: /registration/
Disallow: /search/name-text
Disallow: /search/title-text
Disallow: /find
Disallow: /find$
Disallow: /find/
Disallow: /tvschedule
Disallow: /updates
Disallow: /watch/_ajax/option
Disallow: /_json/video/mon
Disallow: /_json/getAdsForMediaViewer/
Disallow: /list/ls*/_ajax
Disallow: /list/ls*/export
Disallow: /*/*/rg*/mediaviewer/rm*/tr
Disallow: /*/rg*/mediaviewer/rm*/tr
Disallow: /*/mediaviewer/*/tr
Disallow: /title/tt*/mediaviewer/rm*/tr
Disallow: /name/nm*/mediaviewer/rm*/tr
Disallow: /gallery/rg*/mediaviewer/rm*/tr
Disallow: /tr/
Disallow: /title/tt*/watchoptions
Disallow: /search/title/?title_type=feature,tv_movie,tv_miniseries,documentary,short,video,tv_short&release_date=,202
Disallow: /name/nm*/filmotype/*
Disallow: /user/ur*/ratings
Disallow: /user/ur*/checkins
Disallow: /_json/*

User-agent: Baiduspider
Disallow: /list/*
Disallow: /user/*

User-agent: bingbot
Disallow: /showtimes/location/*
```

Figure 5: IMDb robots.txt document.

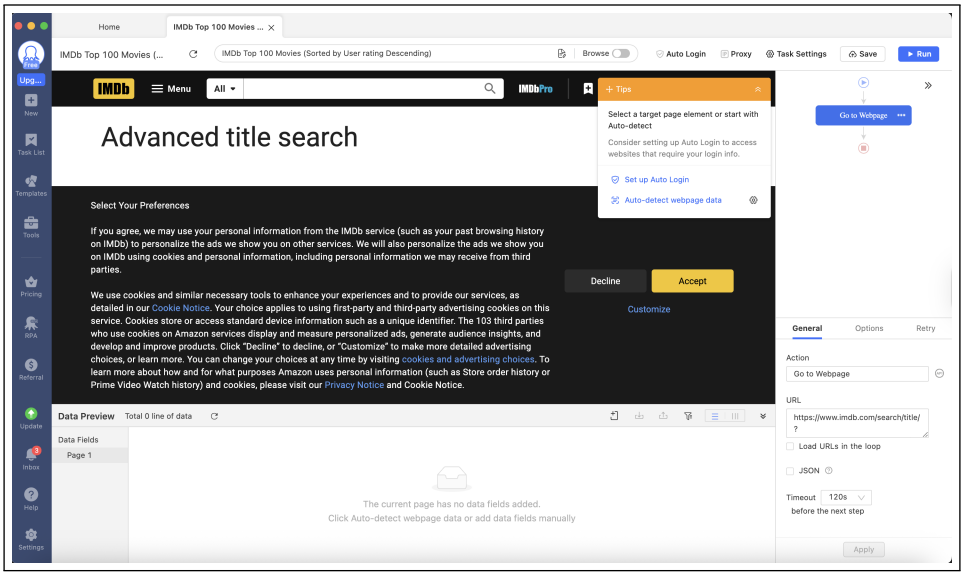

Figure 6: Link opened in the Octoparse browser.

Figure 7: Results of auto-detect feature.

our link into the box. When then click “Save”, which opens up the URL in the Octoparse browser (Figure 6)

As we can see, there is a pop-up in the upper right corner which asks if we want to “Auto-detect webpage data”. Clicking this will start a scan of the page where recurring information elements are detected. Since we are working with a list, the likelihood of it being auto-detected is quite high, and we can make use of the function in order to generate our initial list of URLs. However, once running the scan, the preview indicates some issues. We have only collected information for 50 items, and the title for each film also includes its ranking in the list (Figure 7).

We therefore click the “cancel” button in the auto-detect window, and inspect the site manually. The issue seems to be the inclusion of a button for expanding the list at the bottom of the page, named “50 more”, which is why the web scraper only provided us with 50 lines of data. Clicking the button in the Octoparse browser brings up a dialog box where we can choose the action we want to perform (Figure 8).

Selecting “Click button” adds a “Click item” action to our workflow, and ensures that the list is expanded when the task is later automated. We can now re-try the auto-detection, and see if this solves the issue of the missing movies. Once we have verified that this is the case, we click the “Create workflow” option provided by the dialogue box. The workflow is presented to us on the right side of the screen, and shows that we have now created a loop which collects the information of all items in the list, after having

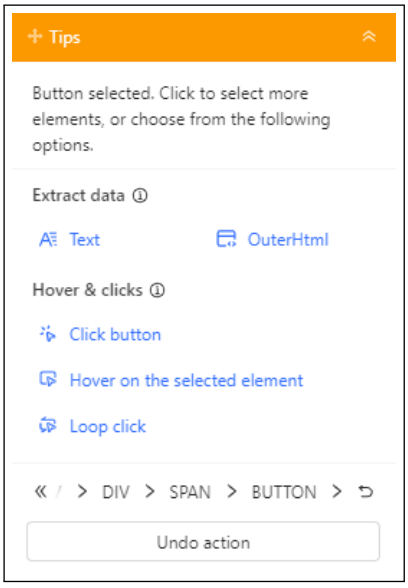

Figure 8: Dialog box resulting from clicking button.

first pressed the "50 more" button. We can now either be happy with the current state of the dataset, or we could click "Select subpage URL" in order to iterate through all of the pages linked in the original list which allows us to select a column in the collected data containing the URL for the subpages.

The latter option will open the first URL in the list, and allows us to select which elements we want to collect from the page. Hovering an element will highlight it in blue, enabling us to see what is included. If we start by clicking the element containing the title, the dialog box asking us which action to perform will appear again. Choosing "Text" will collect the text content of the element, i.e. the title (Figure 9). We then repeat this for the elements containing the short description, the year of release, and the user ranking.

The preview data should now show our selected pieces of information. We can also rename our columns here, to make sure that we know what ends up where in the final output. Once we are happy with the preview data, we click the "Run" button in the top right corner of the window. This will provide us with the different option available for running the task. For this example, we can select "Standard Mode" on our own device. Doing so will bring up a window that allows us to inspect the data as it is being collected (Figure 10). This helps us see if any issues occur without having to run the entire process first.

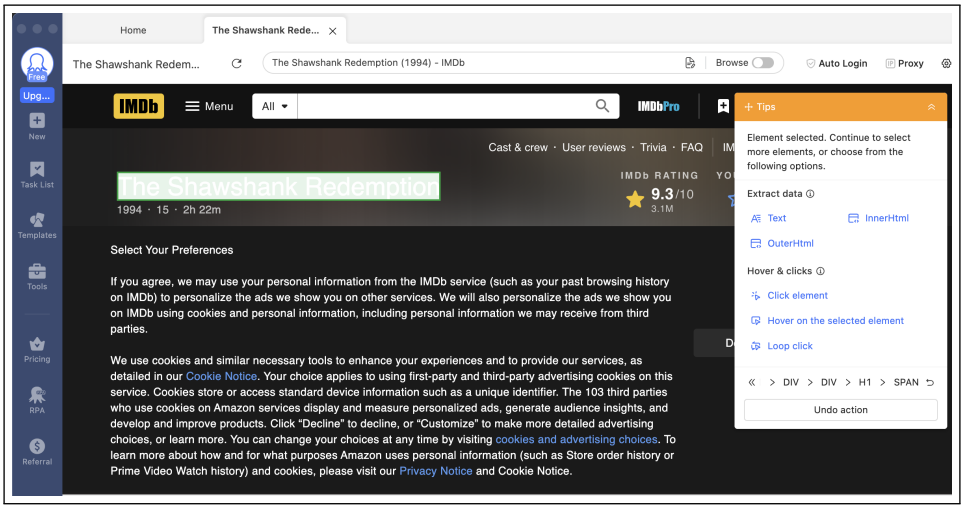

Figure 9: Title selected and text content extracted.

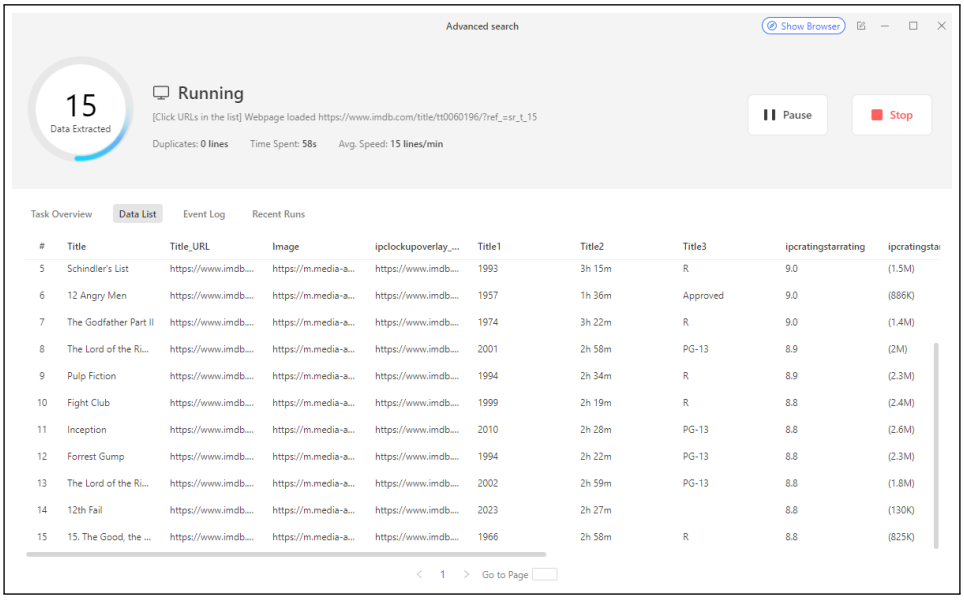

Figure 10: Window allowing for inspection of running task.

Once finished, we can export our data in our format of choice. For this walkthrough, we will choose Excel so that we can easily inspect the results. As we can see, we have created a few duplicate columns by selecting information that was caught by auto-detect in the initial list in our manual selection of elements on each individual page.

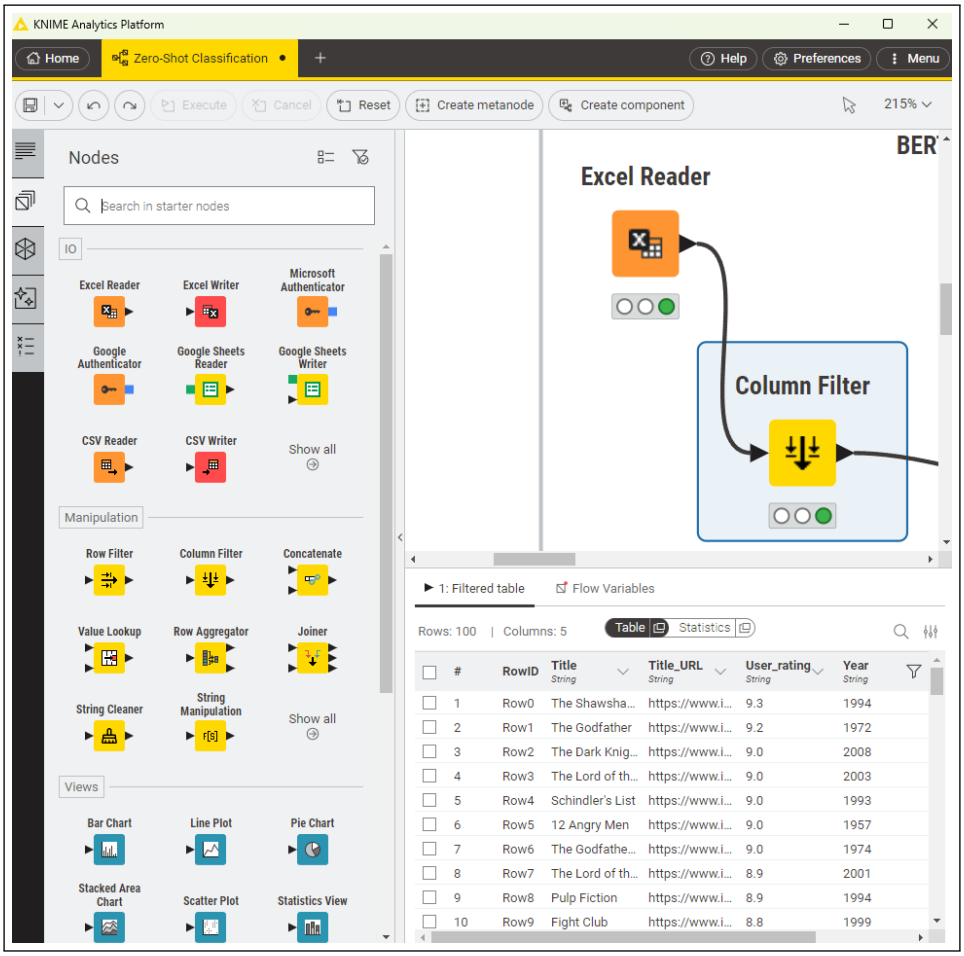

Figure 11: Making the IMDb dataset available in KNIME and removing irrelevant columns.

6.2 *Analysing the Data with KNIME*

We start by opening up a new KNIME workflow, and placing down an "Excel Reader" node so that we can make use of our new IMDb dataset. We then connect our Excel reader to a "Column Filter" node, which will allow us to remove columns that we are not interested in (Figure 11). With these in place, our data is now ready for analysis with KNIME.

We are going to start by performing zero-shot sentiment analysis using a BERT model. In order for us to be able to make use of the nodes related to BERT models, we need to install the Redfield BERT extension for KNIME. Once the extension is installed, we select the "BERT Model Selector", which

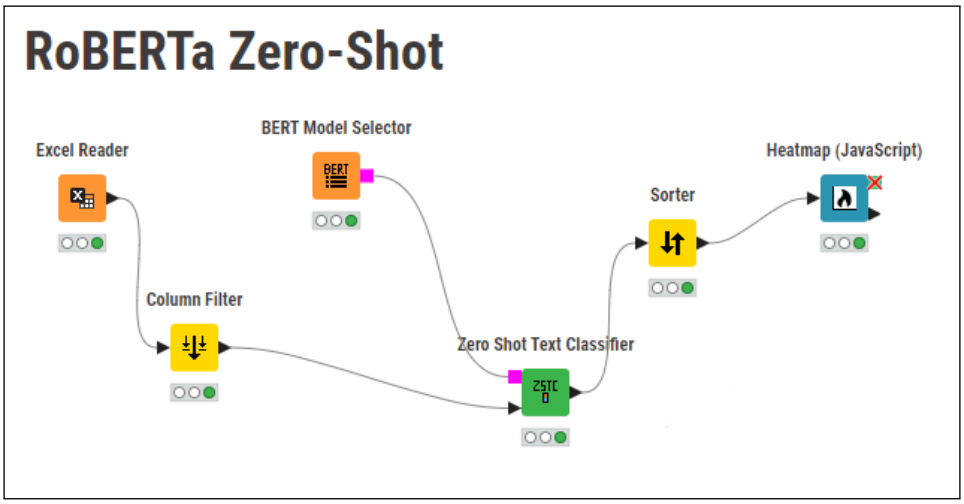

Figure 12: Zero-shot sentiment analysis workflow.

will allow us to select the specific model we want to use. The node allows for the use of a model from HuggingFace, which is a large repository of different models and datasets. In our case, we are going to make use of a model called RoBERTa[2]. We then connect both the model selector and the column filter to a "Zero Shot Text Classifier" node (Figure 12).

Note that the inputs of the "Zero Shot Text Classifier" node are keyed, with one input for the data table and one for the selected model. By right-clicking the "Zero Shot Text Classifier" node, we are able to configure the specific task we want to carry out. In our case, this means selecting the column corresponding to the text we want to analyse, setting up the labels for the classification task, and providing a hypothesis template to indicate the nature of the task. The labels should function as answer alternatives for the question posed by the hypothesis. The configuration for sentiment analysis in the example workflow is shown in Figure 13.

This will add a set of columns to our dataset which will indicate the predicted sentiment, as well as the probability for said prediction of Positive or Negative. We then pass the data along to a "Sorter" node, which allows us to sort the data table according to the predictions. In order to get get a visual overview of the results, we finish the segment of the workflow with a heatmap visualization. In Figure 14, the heatmap is showcasing the probabilities of a Positive label for each of the films.

2 https://huggingface.co/typeform/roberta-large-mnli

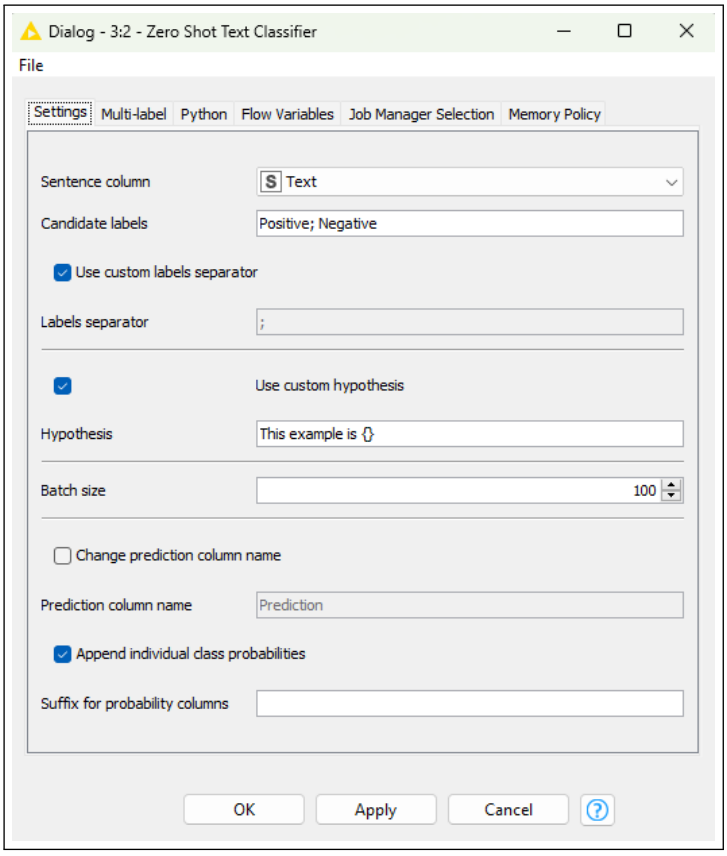

Figure 13: Configuration of "Zero Shot Text Classifier" node for sentiment analysis.

Having performed the sentiment analysis, we then connect our data to the preprocessing stage of the LDA topic modelling section of the workflow. As previously mentioned, preprocessing is a key component for topic modelling, and it is important to consider each step in the context of each project's goals and aims. The preprocessing stage in the example workflow begins by changing the data format, so that it fits with the expected input of the rest of the process.

By using the "Strings to Document" node, we reformat each line of our dataset to a text document. These documents are then preprocessed to remove punctuation, lowercase all words, and remove stop words. Finally, the texts are lemmatized so that each word is converted into its dictionary form. The lemmatization makes use of a SpaCy model (Honnibal & Montani 2017), which is imported through a model selector node similar to the BERT

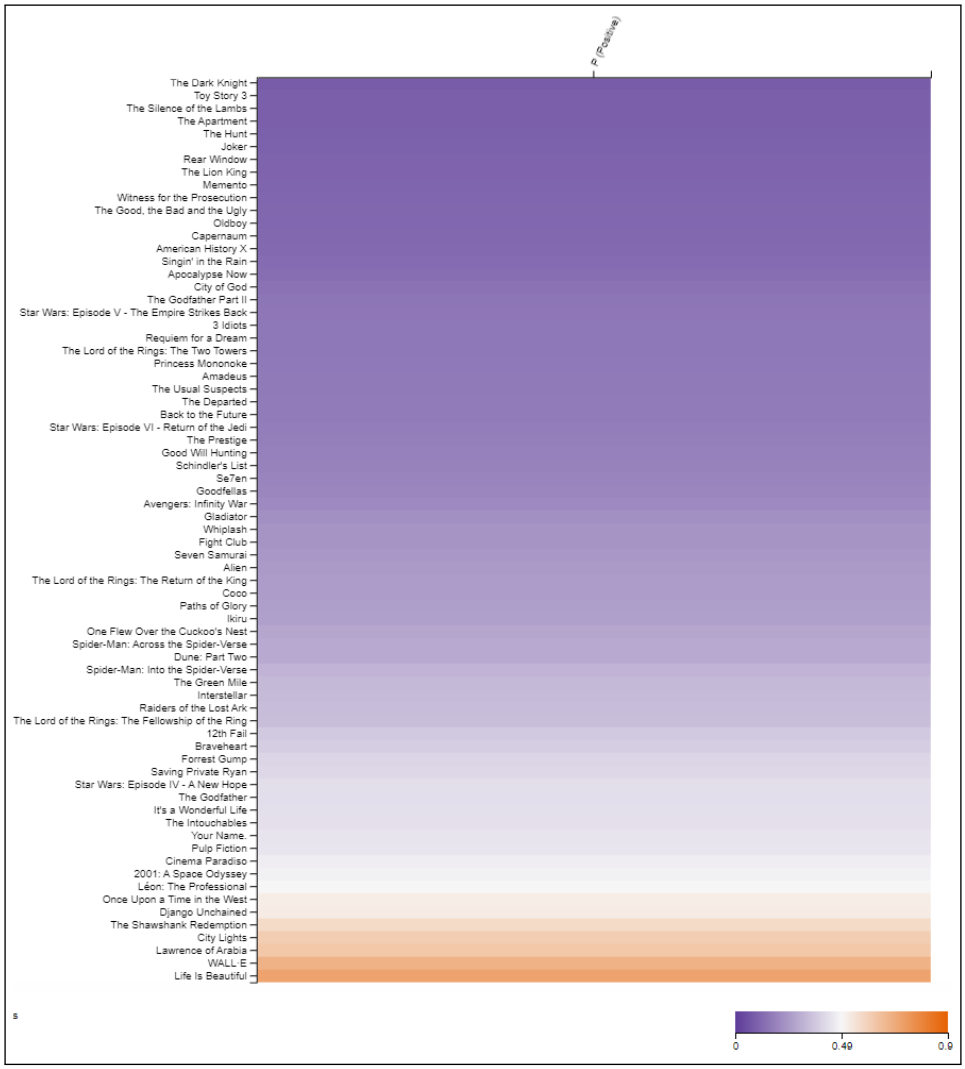

Figure 14: Heatmap showing the probabilities of a Positive label for the IMDb top 100.

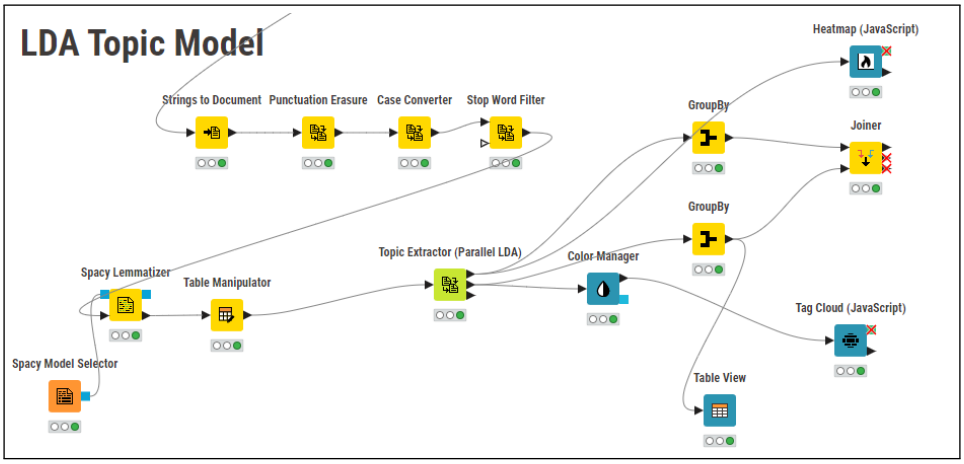

Figure 15: Preprocessing before topic modelling.

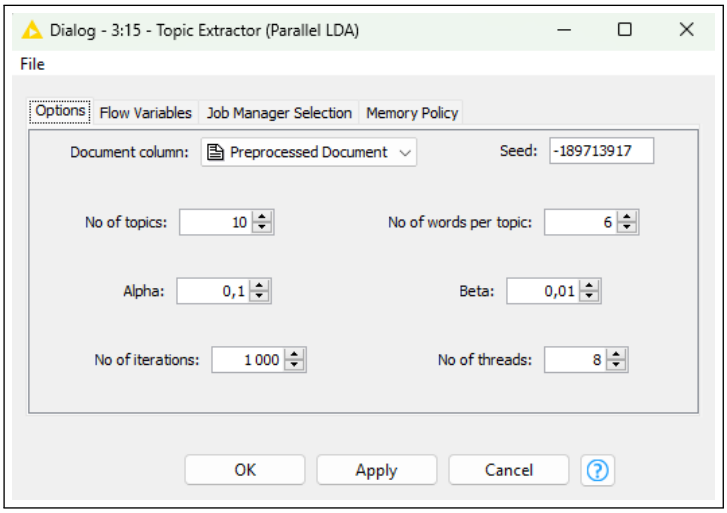

Figure 16: Configuration options for the "Topic Extractor (Parallel LDA)" node.

model selector used for sentiment analysis (Figure 15). As SpaCy models are language-specific, as well as text type-specific to a certain extent, we have to consider the nature of our sample when making this decision. The preprocessed documents are then fed into the "Topic Extractor (Parallel LDA)" node for the actual topic modelling.

In order to run the "Topic Extractor" node, we need to perform some configuration steps (Figure 16). As previously discussed, the LDA method of topic modelling requires us to set a number of topics and the number of

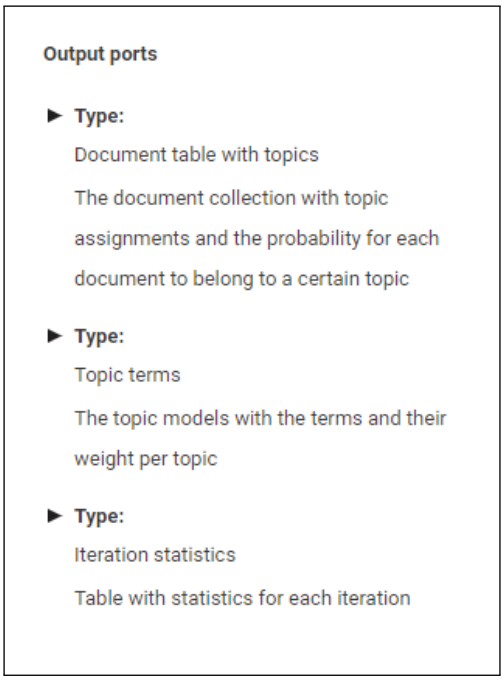

Figure 17: The outputs of the "Topic Extraction (Parallel LDA)" node.

terms per topic before being applied. For our example workflow, the number of topics have been set to 10 and the number of terms to 6. However, feel free to experiment with these settings in order to get as meaningful a result as possible. It is also important to note the column selector at the top of configuration window, as this is where we indicate the column in which we have saved our preprocessed documents.

Once the the node has been executed, we can start looking at the topics we have produced. Here it is important to be aware of the different outputs of the node (Figure 17). The three outputs of the node each produce a different set of information from the process, such as a table of assigned topics with correlating documents, information about the terms included in each topic, and the statistics of the actual extraction process. The input and output specifications are found on the left of the window whenever a node is selected.

These different outputs are each going to inform our understanding of the dataset in a different way. Moving from top to bottom, we start by dealing with the topics assigned for each document. As the internal topic distribution for each document is described numerically, we can visualize

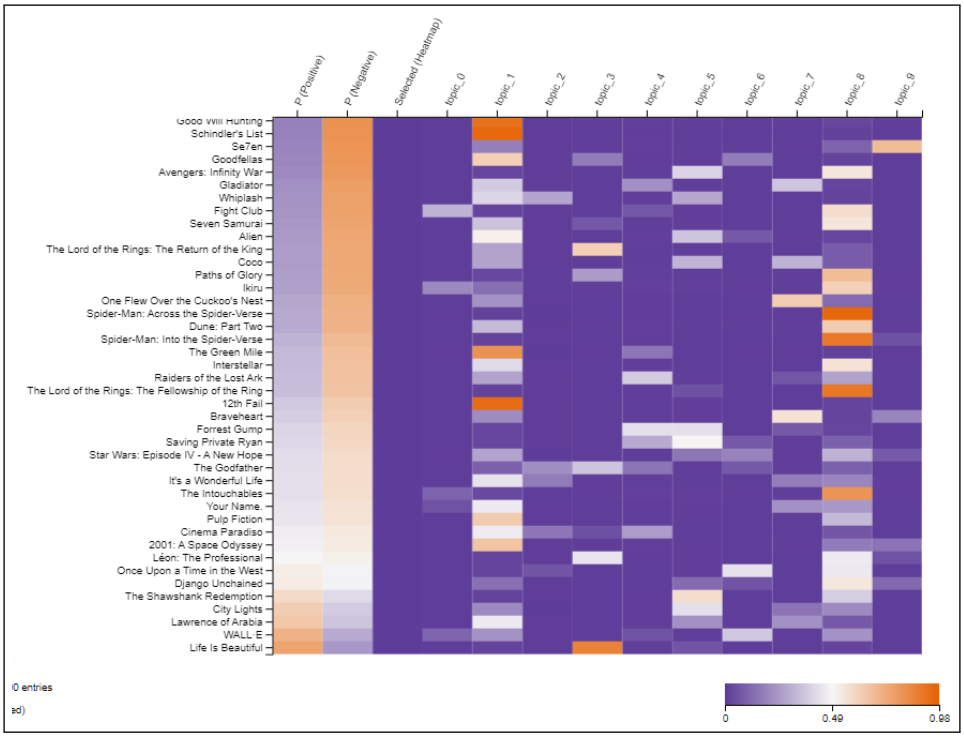

Figure 18: Topic distribution in the IMDb top 100 dataset visualized as a heatmap.

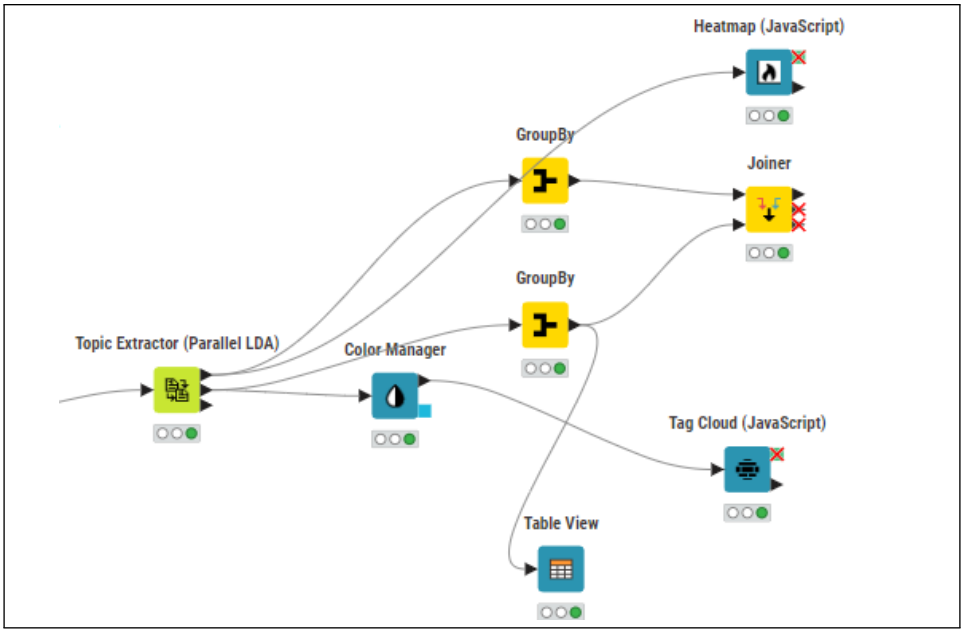

Figure 19: Workflow making use of the outputs of the "Topic Extraction (Parallel LDA)" node to create visualizations and combined data tables.

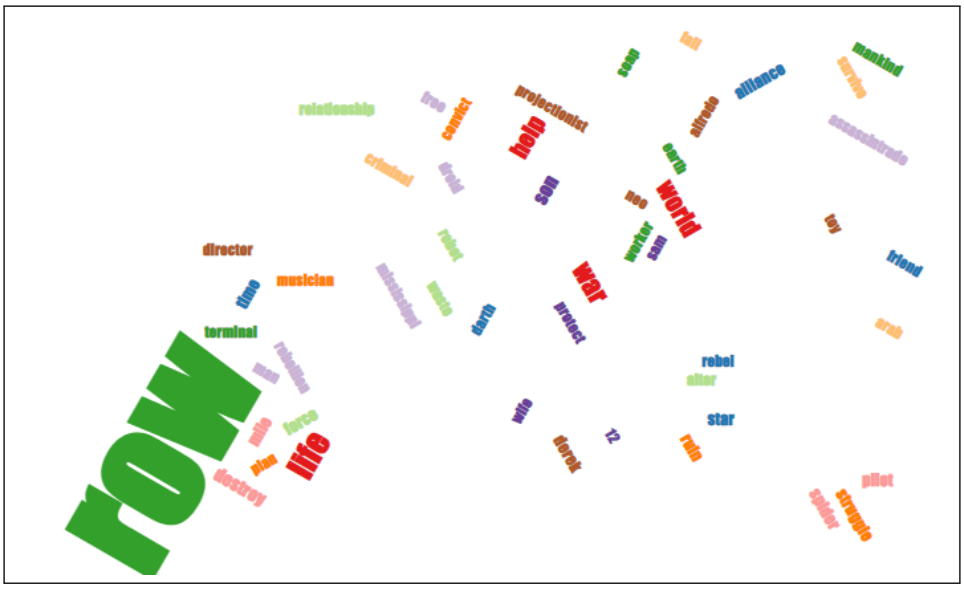

Figure 20: Word cloud created from the topic terms using the "Tag Cloud" node combined with the "Color Manager" node

them as a heatmap (Figure 18). This also allows us to see how topics overlap within individual documents.

We can also combine the top and middle output in order to create a data table containing the documents, their assigned topics, and the terms for the assigned topic. This is done by grouping both outputs according to a shared column, and then joining the two resulting tables together.

We can also make use of the middle output, containing the terms and their weights within the topic, to create a word cloud based on the different terms used. This can serve as an overview of the content of the topics extracted from the dataset. We start by using a "Color Manager" node to assign a consistent color for each of the 10 topics. While this could be done automatically by some of the visualization nodes, the inconsistent use of colors that often result can create ample confusion when several different visualizations are provided based of the same dataset.

After assigning the colors for the topics, we simply pass the output of the "Color Manager" to the "Tag Cloud" node (Figure 19), which creates a word cloud visualization (Figure 20). The configurations for the "Tag Cloud" node allow us to play around with the number of terms included, the sizing based on the weight for the topics, and some other settings that can help us make the visualization as efficient as possible. The word cloud

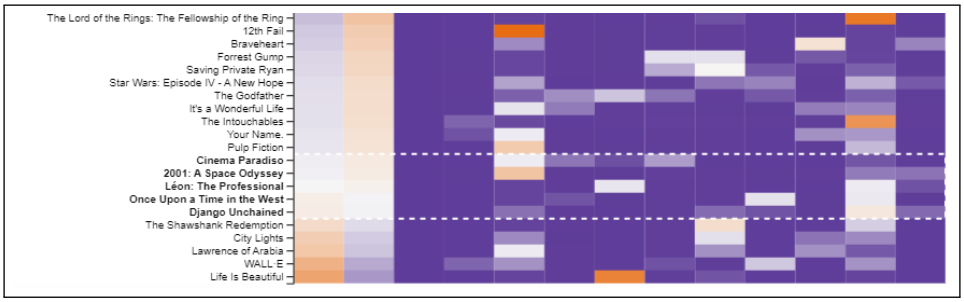

Figure 21: Showcasing a data selection made in the view of the heatmap visualization.

1: Image port ▶ 2: Output table Flow Variables

Rows: 100 | Columns: 9 Table Statistics

RowID	Title String	Title_URL String	User_rating String	Year String	Text String	Prediction String	P (Positive) Number (dou...	P (Negati... Number (dou...	Selected ... Boolean value
Row27	Spirited Away	https://www.i...	8.6	2001	During her fa...	Negative	0.02	0.98	true
Row32	Harakiri	https://www.i...	8.6	1962	When a ronin ...	Negative	0.022	0.978	false
Row84	Eternal Sunsh...	https://www.i...	8.3	2004	When their rel...	Negative	0.023	0.977	true
Row92	Toy Story	https://www.i...	8.3	1995	A cowboy doll...	Negative	0.023	0.977	false

Figure 22: Output data table of the heatmap visualization where current selection is indicated by Boolean value in an appended column.

is also a good way of inspecting the results to find terms that need to be appended to the stop word list. While the stop word list is initially used for removing function words, it can also be used to remove words that either bring nothing of value to the topics or are too frequent amongst the topics, resulting in noisy output.

6.3 *Using Visualizations to Filter*

While there are many different nodes available for different types of visualizations in KNIME, and even more through the use of extensions, the ones used for the example workflow were chosen as they allow for the filtering of data through interacting with the visualizations. In the case of the heatmaps, a selection can be made through clicking "Mouse Mode: Select" in the upper right corner of the view, and then selecting the items of interest. Multiple items can be selected by holding Ctrl (Windows) or Command (Mac) while clicking. Chosen items will be indicated by bolded labels and a dotted line (Figure 21).

Selected items are indicated as such in-place, meaning that the selection does not immediately filter the data. Instead, the use of the interactive visualizations add an additional column where a Boolean value (true or

false) indicates whether or not a specific row has been indicated as selected or not in a given visualization (Figure 22). While this might initially seem cumbersome from the perspective that an additional "Row Filter" node would have to be added to the workflow in order to actually filter the data set, there are benefits as well.

By indicating the selection in place, we have added a new meaningful component to our dataset, which we can make use of further down the line. For instance, it might be useful to know which items were selected based on either sentiment, topic, or both.

The tag cloud offers the same functionality as the heatmap, although selection there is made based on the inclusion of the term selected in the cloud. However, here it is important to keep the input in mind, as the selection column is appended to whatever data table is provided for the node. In the case of the tag cloud used in the example workflow, the input data table only contains the topics and terms. If we are intending to use it for item selection before moving on to further analysis, we might benefit more from using the table produced by the "Joiner" node, as that contains both the terms, topics, and the preprocessed documents.

7 *Concluding remarks*

This chapter has explored the application of low-code tools, specifically Octoparse and KNIME, for web scraping and text analysis within the context of the CICuW project. By demonstrating an accessible workflow that includes data collection, preprocessing, and computational text analysis, we have highlighted how these tools can lower the barrier to entry for researchers with limited programming experience while still enabling sophisticated analytical approaches. The use of zero-shot sentiment analysis and topic modeling with BERT and LDA, respectively, underscores the potential of such methods to generate meaningful insights into large-scale digital discourse.

Despite the advantages of low-code tools in democratizing computational methodologies, we have also emphasized the limitations and challenges they introduce. The "closed-box" nature of such tools can obscure methodological transparency, making it crucial for researchers to critically engage with the processes and assumptions underlying the analyses. Ethical considerations remain a significant concern, particularly regarding web scraping and online data collection, where informed consent is not always straightforward. The adoption of frameworks such as the Association of Internet Researchers' ethical guidelines serves as a necessary step in ensuring responsible research practices. Furthermore, while automated tools can efficiently generate initial

insights, the necessity of combining them with qualitative approaches for contextual interpretation remains paramount, as demonstrated in the CICuW project. Through combining quantitative approaches and qualitative content analysis, the pilot study shows that far-right digital information activities demonstrate a view of public librarians as leftist and as potential threats to the nation. The content analysis of comments from the most commented article, an article about a proposed law requiring public servants to report unauthorized immigrants, indicates a new form of political conflict between national policy and local politics (cf. Harding 2022) that can be connected to the growing national influence of the Sweden Democrats, a far-right party known for nationalism and anti-immigration politics (Dal Bo' et al. 2022).

Ultimately, the methodologies outlined in this chapter illustrate the potential of mixed-methods research in digital humanities, leveraging computational techniques to enhance qualitative inquiry. At the same time, it is important to note that while quantitative methods are necessary when analyzing large amounts of data, qualitative approaches are needed to provide nuanced and contextual understanding. This is particularly important when dealing with contested topics and material rich with irony and sarcasm. By integrating text analysis with exploratory visualization, researchers can identify key themes and focal points for deeper investigation, allowing for a more nuanced understanding of online discourse. As computational tools continue to evolve, future research should aim to refine the balance between accessibility, methodological rigor, and ethical responsibility, especially in cases where closed-box techniques are applied.

References

Berthold, Michael R., Nicolas Cebron, Fabian Dill, Thomas R. Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Kilian Thiel & Bernd Wiswedel. 2009. KNIME - the Konstanz information miner: Version 2.0 and beyond. *ACM SIGKDD Explorations Newsletter* 11(1). 26–31. DOI: 10.1145/1656274.1656280.

Blei, David M., Andrew Y. Ng & Michael I. Jordan. 2003. Latent Dirichlet allocation. *The Journal of Machine Learning Research* 3. 993–1022.

Carlsson, Hanna, Fredrik Hanell & Joacim Hansson. 2022. "det känns som att jag bara sitter och väntar på att det ska explodera" - politisk påverkan på de kommunala folkbibliotekens verksamhet i sex sydsvenska regioner. *Nordic Journal of Library and Information Studies* 3(1). 26–43. DOI: 10.7146/njlis.v3i1.131852.

Dal Bo', Ernesto, Frederico Finan, Olle Folke, Torsten Persson & Johanna Rickne. 2022. Economic and social outsiders but political insiders: Sweden's populist radical right. *The Review of Economic Studies* 90(2). 675–706. DOI: 10.1093/restud/rdac037.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee & Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, 4171–4186. Minneapolis, MA. DOI: 10.18653/v1/N19-1423.

franzke, aline shakti, Anja Bechmann, Michael Zimmer, Charles Ess & The Association of Internet Researchers. 2020. *Internet research: ethical guidelines 3.0*. https://aoir.org/reports/ethics3.pdf.

Hanell, Fredrik & Pernilla Jonsson Severson. 2023. An open educational resource for doing netnography in the digital arts and humanities. *Education for Information* 39(2). 155–172. DOI: 10.3233/EFI-230024.

Harding, Tobias. 2022. Culture wars? The (re)politicization of Swedish cultural policy. *Cultural Trends* 31(2). 115–132. DOI: 10.1080/09548963.2021.1971932.

Honnibal, Matthew & Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Hsieh, Hsiu-Fang & Sarah Shannon. 2005. Three approaches to qualitative content analysis. *Qualitative Health Research* 15(9). 1277–1288. DOI: 10.1177/1049732305276687.

Hutto, C. & Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. *Proceedings of the International AAAI Conference on Web and Social Media* 8(1). 216–225. DOI: 10.1609/icwsm.v8i1.14550.

Jones, Steven E. 2016. *Roberto Busa, S. J., and the emergence of humanities computing: The priest and the punched cards*. 1st. Routledge. DOI: 10.4324/9781315643618.

Kozinets, Robert. 2020. *Netnography: The essential guide to qualitative social media research*. 3rd. Thousand Oaks, CA: Sage Publications.

Octoparse. 2024. *Octoparse 8*. https://octoparse.com/.

Scrivens, Ryan, Garth Davies & Richard Frank and. 2020. Measuring the evolution of radical right-wing posting behaviors online. *Deviant Behavior* 41(2). 216–232. DOI: 10.1080/01639625.2018.1556994.

Selenium. 2024. *Selenium Web Driver*. https://selenium.dev/.

Shneiderman, Ben. 1996. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings 1996 IEEE symposium on visual languages*, 336–343. DOI: 10.1109/VL.1996.545307.

Tyrkkö, Jukka & Daniel Ihrmark. 2024. Low-code data science tools for linguistics. In Steven Coats & Veronika Laippala (eds.), *Linguistics across disciplinary borders: The march of data*, 40–66. Bloomsbury. DOI: 10.5040/9781350362291.0008.

Wing, Jeannette. 2006. Computational thinking. *Communications of the ACM* 49(3). 33–35. DOI: 10.1145/1118178.1118215.

List of abbreviations

AoIR	Association of Internet Researchers
BERT	Bidirectional Encoder Representations from Transformers
CICuW	Cultural Institutions and the Culture War
DOM	Document Object Model
HTML	HyperText Markup Language
IMDb	Internet Movie Database
IRE	Internet Research Ethics
JSON	JavaScript Object Notation
KNIME	Konstanz Information Miner
LDA	Latent Dirichlet Allocation
RoBERTa	Robustly optimized BERT approach
VADER	Valence Aware Dictionary and sEntiment Reasoner

Corresponding author

Daniel Ihrmark
Department of Cultural Sciences
Linnæus University
daniel.o.sundberg@lnu.se