
Production Systems and Information Engineering
Volume 13 (1), pp. 95–111.
https://doi.org/10.32968/psaie.2025.1.8

EXPLORING EFFICIENT NEWS AGGREGATION DEVELOPMENT FOR THE HUNGARIAN LANGUAGE

BENCE ÓNODI[1]
University of Miskolc
Hungary Institute of Information Technology
onodibenc@gmail.com

ANITA AGÁRDI
University of Miskolc
Hungary Institute of Information Technology
agardianita@iit.uni-miskolc.hu

Abstract. In recent years, the rise of digital media has fundamentally transformed information consumption habits. The mass of content generated by social platforms and online news portals makes it increasingly difficult to obtain information from reliable, objective sources. The spread of fake news, the dominance of radical views and click-oriented media phenomena distort public discourse, making critical thinking difficult. The article presents the development of a Hungarian-language news aggregation system that not only collects news from different sources, but also helps their interpretation and transparency with its analytical functions. The goal is to create a technological solution that supports users in the conscious and critical processing of information, contributing to balanced information in the Hungarian-language media space.

Keywords: *news aggregation system, digital media, Hungarian language*

1. Introduction

In recent decades, the spread of digital media has taken on unprecedented proportions. The endless abundance of content provided by social media and the constantly updated news portals have revolutionized the flow of information, with many advantages and disadvantages. One of the most significant negative

[1] This article summarizes the author's BSc thesis entitled *Magyar nyelvű híraggregáló webalkalmazás fejlesztése* submitted to the University of Miskolc in 2025.

consequences is that it is more difficult for content consumers to get information from reliable and objective sources. The hidden biases behind the news often distort its content and presentation, and further complicate its understanding. There is a worldwide trend that extremist and radical trends have an increasingly greater reach than moderate ones. This is largely due to the operating mechanism of online media, where click-baiting, attention-grabbing titles dominate. Internet content creators often use highlights at the beginning of their videos that immediately capture the attention of viewers. This practice favors the rapid spread of fake news and unfounded information. All this distorts the media and public discourse, influencing even those who would otherwise be balanced opinion formers. These phenomena make technological solutions that help navigate the often opaque information space particularly relevant. The article aims to provide a solution to a current and complex problem, as fake news and manipulation pose significant challenges to people around the world, and Hungarian-language online media is no exception. The news aggregator site is not only used to collect news, but also helps readers with analyses to make it easier to process information critically and consciously.

It is important to emphasize that the aim of the article is exclusively technological implementation, and neither the expression of specific news sites nor political positions in general is among its objectives. The presentation of the applied methods and results also serves this neutral, analytical purpose.

1.1. Role and operation of news aggregator systems

A news aggregator is a system, that collects content from different sources – for example, online newspapers, blogs or other media – and then groups it according to specific criteria and displays it on a common interface. The aim of such systems – which are typically available in the form of web or mobile applications – is to make news quickly and easily accessible to those who want to be informed about the events taking place around them from multiple locations. These programs centralize news sources, saving time and energy, and contributing to a more objective follow-up of the news.

There are many news aggregators on the market, but they can basically be classified into two large groups.

The technical process of the two groups is often similar, but they differ in their principle of operation. One group includes edited format news aggregators, in which different journalists select the content for us. Such sites prioritize quality over quantity, as this is a method that requires human intervention, and therefore often use analyses or various opinion-forming extracts to enhance the presentation of news. The other group places greater emphasis on quantity and hosts comparative analyses rather than individual analyses. These systems do not use manual intervention and represent the added-value service in a different form. Among them, passive systems can be observed – these only collect and display information, thus providing space for comparison – and active systems, which also

provide an interactive user experience. The latter often have different filtering options (for example, by topic, date or source), and some systems even support the compilation of a personalized news feed. Such interactive platforms can be compared to widespread social media applications, but they mostly actually place the emphasis on information. Nowadays, users can choose from a wide range of news aggregator applications, so they can easily find a solution that suits their needs. The use of advanced topic categorization is increasingly common in modern services, and with the rise of artificial intelligence, functions that previously required manual intervention and are mostly based on language models or machine learning have also appeared. Such functions include sentiment analysis, which attempts to describe the mood of articles, or fake news filtering, which is of particular importance in addition to algorithms that are optimized to capture users' attention. These innovations significantly increase the added value of aggregators, which is typically not provided by online newspapers themselves.

1.2. Web data collection techniques

The development of web data collection techniques has laid the foundation for the spread of news aggregator systems, since data collection is a key component that allows to manage huge amounts of information in a structured way. Among the services of online newspapers, the Really Simple Syndication (RSS for short) [1] function, which provides a short summary of the content that appears on the page and is updated frequently.

In addition, another widespread format is Atom [2], which serves a similar purpose: offers updates about web content in a structured form, but implements more advanced metadata management, greater customization, and stricter XML-based standards compliance.

Both RSS and Atom formats allow news aggregator systems to easily access the links, titles, publication dates, and often cover images of articles.

One of the most common technologies is web scraping, which aims to extract structured data from HyperText Markup Language (HTML)-based content available on the Internet in a programmatic manner. This process allows us to extract valuable information from pages intended for human reading in an automated manner. Since web scraping involves analyzing the HTML structure of websites, it is primarily useful in cases where the given page does not provide public API access. Due to the proliferation of pages on the Internet, many programming languages offer solutions for this with different packages, such as:

- BeautifulSoup: a popular Python library used for structured data extraction from HTML and XML files. It allows for simple and efficient traversal and search of the DOM tree of pages, as well as extraction of the desired data elements. [3]
- Puppeteer: A Node.js-based library that allows programmatic, headless control of the Chrome or Chromium browser. [4]
- Selenium: An automated browser control tool that allows for managing interactive elements of websites and accessing dynamically loaded content. [5]

1.3. Basics of Natural Language Processing

Natural Language Processing (NLP) combines linguistics and computer science. Its goal is to enable machines to recognize, understand, and generate human language in written or spoken form using machine learning. Research into the method laid the foundation for the era of artificial intelligence that is now considered natural. Services and tools that use this method have become an essential part of our lives today, as they power the largest search engines, operate automated customer services, and enable digital assistants. [6]

- Word segmentation (tokenization): Tokenization is a process in which text is divided into words. This process results in a list of words (tokens) and their positions in the text.
- Lemma formation: By removing suffixes, the dictionary (basic) form of the words is obtained. The simplified form is called a "lemma".
- POS tagging: This task determines the part of speech of the words in the sentence. It is common for a word to have multiple part of speech roles, such as the word "vár", which can be a noun ("„Megújul a vár.") or a verb ("Villamosra vár."). The part of speech analysis of such cases requires taking the linguistic context into account.

2. System specification and requirements

2.1. Purpose of the system

The purpose of the developed system is to automatically collect news from online sources of Hungarian-language media and then analyze them with a self-developed language processing process. The goal is to make it clear to visitors to the page what kind of emotional charge can be recognized in each source or given article. In addition, the application should be able to recognize political or other public figures from the content, thus helping the reader to compare the mood in which the given person is mentioned in different sources.

2.2. Functional requirements

The focus of the system's operation is the automatic collection, processing and user-friendly display of news. The main functional requirements to be met by the system are listed below.

2.2.1. Automated news gathering

The basic function of the system is to automatically download newly published articles from selected news portals at predetermined intervals. This process is essential for the operation of the system, since in the absence of automatic news gathering there would be no content available for further processing, thus neither analysis nor visualization could be implemented. The system can only fulfill its

purpose if content updates are continuous and reliable.

2.2.2. Structured storage of articles

Relevant metadata for downloaded news – such as the title of the article, date and source URL-together with the full text of the article in a relational database are stored. This solution allows efficient querying, the also filtering content and post-processing. The cover images belonging to articles are not placed directly in the database but in the server-side file system. The database only stores the path of the images that is based on which the system can retrieve and display the given image through a separate serving layer on the user interface. This approach reduces the load on the database as well as provides fast and scalable access to files.

2.2.3. Identifying content

After processing the text of the articles, the system is able to automatically identify persons in the writings are an entity recognition (Named Entity Recognition, NER [7]) module. Texts related to persons thus identified in separate sentiment analysis fall through that allows you to examine what a particular person appears in an emotional context in each news sources. It can aggregate results through statistical statements such as some political actors which medium is represented by what frequency and what emotional filling.

2.2.4. Sentiment analysis

The system automatically assigns a sentimental [8] value to all the texts related to all articles, including the specific person, to the specific person. The purpose of scripture analysis is to explore the emotional charge of the text and thus give an idea of the communication tone of the news source. The sentiment analysis model performs a separate classification for each token (word or word fragment), classified as one of five categories. Classes can be assigned to the labels returned by the model with the following numeric values:

- label_0: −2 (very negative),
- label_1: −1 (negative),
- label_2: 0 (neutral),
- label_3: +1 (positive),
- Label_4: +2 (very positive).

The screed value of the full text is given by the arithmetic average of the numeric classes assigned to the tokens. This number is located on a continuous scale that can be redeemed to text categories using the following intervals:

- [−1.5; −0.5 [range: very negative,
- [−0.5; 0 [range: negative,
- 0: neutral,
- [0; 0.5] range: positive,
- [0.5; 1.5] Range: Very positive.

This approach allows us to measure the emotional shades more subtle, as it not only assigns a class to the whole text, but also takes into account its internal diversity. The model also associates a confidence score for all token classification. These values can be used as required when calculating the aggregated sentiment, but the default method is the non-weighted averaging. The resulting average sentiment is the basis for further statistical statements and visualizations.

2.2.5. Display articles in the web

More modes can be used to display articles on the user interface. Articles from the news portals taken into account get a separate page, and a simplified feed also serves the interests of readers where articles in analyzed media become available on chronological order. In addition, you can filter your date screening on each page and set on a saint scale what you want to read.

Downloaded and processed articles on the user interface are displayed in different ways for easy transparency and targeted browsing. The system provides two main views:

- Source -specific article pages: Each news portal receives a separate page where only articles from a given medium appear.
- Simplified News (FEED): A central view that chronologically classifies all available, analyzed articles, regardless of their source.

For users, additional filtration options are available on each page:

- Date-based screening: It is possible to view articles published during a given period.
- Sentiment screening: An interactive scale allows you to select what emotional articles appear (for example, only neutral and positive content).
- Category filtering: It is possible to display articles in the same group based on the categories assigned to the media.

This structure allows readers to quickly navigate the news and filter information based on their personal interest.

3. System Architecture and Planning Aspects

In order to achieve the goals of a system, it is necessary to develop a comprehensive architecture that reliably handles automatic data collection, processing logic, data storage, and data provision to the client site. This section shows the technological background of the selected technological background, the construction of the server-side file structure, the operation of the processing pipeline, the data storage solutions, and the role of client-side components in the display. The primary consideration of the system was modularity, reusability and easy expansion, which are essential for a long-term sustainable, efficient and well-scale operation.

The architecture is made up of three main parts: the client-side application, the server-side central control unit, and the background processing pipeline. The client transmits the data to the server through HTTP requests, which manages traffic based on appropriate routes and then transfer requests to the processing module if necessary. During the design, the components are separated so that the system could be easily expanded, modified and maintained.

3.1. Server side

The central element of the operation of the web application is server-side logic, which is responsible for controlling background processes, data collection, data processing and service to clients. Server-side components also ensure the reliability of the data flow, the validity of the data and the scalability of the system.

3.1.1. The framework

Node.js [9] is an open source, event controlled, platform independent environment that runs javascript. The platform is based on the Google V8 Javascript engine and follows the "Run JavaScript Everywhere" paradigm, which extends the language to the server page, allowing the development of full web applications using a single programming language. This environment also runs perfectly on Windows, Linux and MacOS, and during its operation, it provides a single process to serve the requests without creating fibers. Thanks to this, it provides a high performance and reliable solution, so it is widespread in the field of IT.

It is supported by its own package manager, NPM (Node Package Manager). NPM [10] is one of the largest packages in the world, providing access to hundreds of thousands of open source modules. The developer community is active, constantly creating new packages, making the system inevitable in modern web development.

A minimalist and flexible framework, the Express.js. [11] is chosen as server-side. Express simplifies the use of Node.js, offers a more transparent and readable code structure, and supports a simple and structured definition of routes. The large and active community provides a plentiful documentation and additional package, which facilitates the development of complex systems.

3.1.2. Structure of the source code for server -side application

When designing a server page, the goal was to separate different functional units and to be organized into well structured modules. The following folder structure is applied:

- Config: Here are the files containing configuration settings. Currently, for example, the Connection.js file, which initializes the modules needed for the database connection and creates the database connection.

- Controllers: The application control logic is located in this folder. The controllers process incoming HTTP requests, calls the necessary services, and then gives the answer to the client.
- Middleware: Intermediate layers between the client and the server logic. Here can be found logging modules, error handlers or features responsible for validating data.
- Router: Folder defining the path routes. This is where the individual endpoints (such as querying articles or screens analysis) will be configured and assigned to the appropriate controllers.
- Scrapers: Each news portal has a separate scraper file, which, with the help of the Puppeteer directory, performs DOM processing and reducing data according to the structure of the given website.
- Services: A folder containing business logic. For example, these services are responsible for processing data, saving it into a database, or communicating with the external scripture API. The controllers call these services.
- Utils: Utils contains for reusable, general auxiliary functions. These include date formation, text cleaning or other logical operations.
- .env: A file containing the environmental variables needed to run a project. For example, storing the availability of the database, API keys or other sensitive information, etc.
- Index.js: The entry point of the application. This is where the Express server is launched, MiddleWare registration and loading routes. This file is the first one that Node.js runs when you start the application.

3.1.3. Data collection

The central element of the research is data collection, as subsequent processing and analysis steps can only be based on the success of this. To do this, a tool is inevitable that is easy to manage, widely used, and its developer community is active and is technologically well fit into the server-side environment. Based on these aspects, the library called Puppeer [12] is chosen, which is part of the Node.js ecosystem and allows the control of Chromium-based browsers, such as Chrome and Firefox, through a high level of API. The layers of the chosen library ensure that we can process the web pages without having to write low -level codes to control the browser. The technological background behind Puppeer — JavaScript programming language and Google development support — is a guarantee that the project remains sustainable, documented and developed in the long run.

The Puppeer operation is based on the Chrome Devtools Protocol, a high level of communication protocol between the browser and the control program. This protocol allows dynamic control of websites, such as loading pages, clicking on buttons, filling forms or taking screenshots. The Puppeer is used not only for data reduction, but also for testing and behavioral level simulation, making it an ideal choice for the automatic data collection component of a news system. [4]

Data extraction process from news sites

During data collection, the goal was to automatically extract relevant information from articles appearing on news sites: the title, the publication date, the content of the articles, and their associated images. The steps of the data collection logic are generally structured as follows:

1. Puppeteer launches a headless browser instance and loads the main page of the given news site or a URL containing a given list of articles.
2. Based on the analysis of the DOM structure of the page, identifying the elements that contain the list of articles (for example, `article` or `div.article-card` tags).
3. Within the given page, iterating over the HTML blocks representing the articles and read from them:
   - the title of the article (from `h1` or `h2` elements),
   - the publication date (for example, time tag or a given class),
   - the detailed link associated with the article,
   - the thumbnail image (image URL).
4. It opens the article pages individually based on the URLs identified in this way and extracts the full text of the article from them.
5. All extracted data is serialized in JSON format and saved to a database.
6. During data upload, the system tries to avoid duplications based on the date and title of the articles (it does not save the previously queried article again).

The above process allows the system to collect fresh data from the news pages even on a schedule (for example, hourly or daily). Different retrieval logics apply to different sources, since the HTML structure is not uniform. Data collection handles the structure of the news pages and the pagination mechanism in a separate module.

3.1.4. Data provision

The web application server provides the necessary data to the clients using REST APIs. The server operates in a Node.js environment with the Express.js framework, and the data exchange is in JavaScript Object Notation (JSON) format. The purpose of the data provision is to query articles, save new articles, and provide various sentiment analysis statistics. The client-side React application communicates with the server via HTTP requests, and data is retrieved using GET, while new data is saved using POST. In its current state, the system does not include user authentication, all endpoints are publicly available in the development environment.

3.2. Data processing process

In modern, high-value news aggregation web applications, simply collecting and storing data is not enough. It is necessary to extract easily interpretable, structured information from raw text data, which helps users navigate and analyze faster. To do this, a data processing pipeline had to be created that encompasses all steps from data download to text analysis. The system implements this process through the close cooperation of several components: a Node.js-based Express.js server, a natural language processing module written in Python, and ensuring data flow between the database and the client.

3.2.1. The data path

After successful data collection, the text content of the downloaded and pre-processed articles must be processed by the system using Natural Language Processing (NLP) techniques in order to draw conclusions from it later from a statistical or reader perspective. JavaScript does not offer a suitable processing unit optimized for the Hungarian language for such tasks, so we had to outsource this layer to a separate, Python-based system. Communication between the two systems is implemented via a REST API, which enables fast and reliable data transmission. The main tasks of the Python component include natural language processing, including: Named Entity Recognition (NLP) and sentiment analysis. For entity recognition, the huSpaCy library is used for the Hungarian language optimization, which is capable of tokenizing texts and recognizing names in a Hungarian context.

huSpaCy [13] is specifically adapted to the grammatical characteristics of the Hungarian language, thus enabling a more precise analysis than the international general models. For sentiment analysis, a transformers library is used, which specializes in handling large language models. The specific sentiment model was the NYTK/sentiment-hts5-xlm-roberta-hungarian [14] model of the Linguistics Research Center, which is based on the XLM-RoBERTa [15] architecture. This model was trained on Hungarian-language texts and distinguishes five sentiment categories: very negative, negative, neutral, positive and very positive. The model is based on text corpora that come from political, public affairs and general news, making it particularly suitable for analyzing Hungarian-language news portals.

We can also find similar examples in international practice: for example, the Ground News [16] platform also uses sentiment analysis and cross-source bias testing to make the political and emotional orientation of news more transparent.

The implemented system has a similar logic, but tuned to Hungarian language specifics. The Python-based NLP component operates as a separate Flask [17] server, which the Node.js-based server accesses with HTTP requests. The client or data collection module does not communicate directly with the NLP module, only the server API calls the appropriate endpoints.

3.3. Data Storage

There are several data storage solutions available for web applications, including relational databases, NoSQL databases, and file system-based storage for certain types of data, such as image files. The application has a relational (SQL-based) database because the data is well-structured, interrelated, and can be easily queried using the SQL language.

It is important to note that the system also handles images, but they are not stored directly in the database. The actual image files are stored in the server file system, and the database only contains the file paths to the images in the form of text fields. This provides significant performance and storage benefits because the database is not burdened with large amounts of binary data.

Among the relational databases, the PostgreSQL [18] is chosen, which is one of the most reliable and popular open source database management systems. Its advantages include high scalability, advanced SQL support, transaction management, and an active developer and user community. In addition, it can be integrated with modern backend environments and serves the performance and reliability needs of the web application well.

3.3.1. Details of the application tables

During data modeling, the relationships between the data is developed as clear and easy to query, so ensured the relationships between the different tables with foreign keys. The purpose of the created database structure was to effectively support the operation of the application, and to quickly access and modify data. Most of the data stored in the database is structured text or numerical information, such as article titles, content, author names, categories, and various bias values. The system also handles image files, but they are not stored directly in the database. Instead, the database only records the server-side path to the image files, while the image file itself is located in the server's file system. This approach relieves the database, reduces response times, and allows for more flexible file management.

`article_tags`
This table implements the many-to-many relationship between articles and tags. An article can belong to multiple tags, and a tag can be linked to multiple articles. The `article_id` column is a foreign key that references the primary key of the articles table, while `tag_id` references the identifier of the tags table. The `updated_at` field stores the time the record was last modified.

`articles`
This table stores newspaper article data, including the title, content, URL, image, and publication date. The `id` is the primary key, the `title, content, url`, and `img_url` fields record the textual characteristics of the article. The table also contains three foreign keys: `category_id` refers to categories, `author_id` refers to authors, and `source_id` refers to the sources table. The

published_at and updated_at fields record the time of publication or update, while bias_score contains the bias value of the article.

authors

This table stores data of authors. The id is the primary key, the name field contains the name of the author. updated_at records the time the record was updated, and bias_score records the bias value associated with the given author.

categories

The categories table serves as the basis for the thematic classification of articles. The id is the primary key, name contains the name of the category, and description contains its detailed description. The updated_at field records the time the record was last modified.

sources

This table stores online news sources. The id is the primary key. name describes the name of the source, url describes its accessibility, category describes the orientation of the source (government-related, independent), and updated_at contains the time the record was updated.

person_bias

A table containing the bias values of persons for public figures. id is the primary key, name records the name of the person, and bias_score records the bias value. The source_id field is a foreign key that points to the sources table. updated_at contains the time the record was updated.

tags

A table containing tags, keywords, with which articles can be categorized. id is the primary key, name contains the name of the tag. The source_id field is a foreign key that points to the sources table, and updated_at records the time the record was last modified.

Figure 1. ER diagram of the database

After introducing the role and data content of each table, the ER diagram shown in Figure 1 illustrates the entire structure of the database and the relationships between each entity. The diagram clearly shows how articles, sources, categories, authors, tags, and sentiment values assigned to people are related to each other.

3.3.2. File system and file management

The system handles not only structured data, but also binary files, primarily images. These are not stored directly in the database, but in the server's file system in order to make more efficient use of performance and storage space. Images are saved in the `/public/images` directory, and the paths are stored in text form in the database. Scraper modules automatically generate file names and avoid name conflicts. This approach is flexible, maintainable, and allows images to be managed separately, even when moved to external storage.

4. Implementation and Deployment Process

4.1. Server Side Implementation

During the development of the server, first the data collection modules are created using Puppeteer. The purpose of the scraper was to automatically crawl the pages, collect the titles, publication dates, images and texts of the articles, and then store the data in a structured way. The scraper can handle the pagination of the pages and prevent the saving of duplicate entries.

For this module, REST API endpoints are developed using the Express.js framework, which provide access to the data for the client side. During the development of the APIs, great emphasis on robust error handling is placed, which allows for the appropriate handling of network or database problems.

4.2. Pipeline Implementation

The purpose of the processing pipeline was to clean the collected data, analyze it, and save the results back to the database. HTML tags were removed from the text data, the texts were normalized, and erroneous data was filtered out. For sentiment analysis of the data, a pre-trained Hungarian NLP model is used, which assigned a bias value to each article. The pipeline has a modular structure, and the data processing steps are implemented in separate modules, thus ensuring easy extensibility and maintainability.

4.3. Client side implementation

During the development of the frontend, the goal was to create a clean, easy-to-use user interface. The component-based approach of the React framework allowed for separate, modular implementation of each view (home page, statistics pages). On the home page, newly downloaded articles are displayed with infinite scrolling, in card format. On a separate page, statistical analyses are also available using diagrams, such as the change in average bias values over time.

4.4. Adding a new news source to the system

The application was designed to make integrating news source easy. The scraper modules are located in a separate file and communicate with the backend via a unified interface. The following steps are required to support a new news portal:

1. Create a new scraper module

The first step is to create a new JavaScript file in the scraper folder (for example, `sourceName.scraper.js`). In this file, it is necessary to implement the crawling of the website of the given news portal, the identification of relevant DOM elements, and the collection of the following data using the Puppeteer library:

- article title,
- publication date,
- image URL (if available),
- full text.

The pagination of the page also needs to be handled if articles can be collected from multiple pages.

2. Adapting data saving to the system

The collected data must be stored through the existing database management layer. The system includes a unified `saveArticle()` function, which ensures data validation and structured saving. A unique identifier based on the title and date of the article is used to avoid duplicate records.

3. Adding a new endpoint

To call the scraper module, a new route and its associated controller must be defined in the server-side Express.js application. This requires the following elements:

- a route entry (for example `/get-latest/sourceName`),
- a controller function associated with the route that calls the scraper module.

4. Testing and validation

It is advisable to test the operation of the new scraper by running it manually. It is advisable to check the collected articles:

- whether they are saved to the database correctly,
- whether the pipeline processes them without errors,
- whether they are displayed on the client side as expected.

The scraper operation can then be scheduled using the existing cron-based mechanism. By following the steps described, the integration of a new news source can be done by creating a few files and making minimal changes, without affecting other components of the system. This ensures scalability and long-term expandability.

4.5. Achieved results

The following main achievements were achieved during the development process:

- Data collection: Automatic collection of news articles from two major Hungarian news sources was achieved using Puppeteer-based scraping modules.
- Text processing: Raw data processing, cleaning and the application of a Hungarian-language sentiment analysis model were successfully integrated into the processing pipeline.
- Data storage and API: Data was stored in a structured manner and made available via a REST API.
- Client-side display: A responsive, card-based news reader interface and statistical charts were created to support the user experience.

Approximately a thousand articles were downloaded, processed and sentiment analyzed during the project.

4.6. System limitations and development opportunities

Although the developed system performs its basic functions well, shortcomings

and development opportunities can be identified in several areas.

4.6.1. Data collection sensitivity

The operation of scraping modules strongly depends on the current HTML structure of the pages. In the event of a change in the page structure, the scraper can quickly become faulty, which results in a need for maintenance. In the future, it would be advisable to:

- Apply machine learning-based page structure recognition.
- Introduce content-based scraping solutions.
- Develop or use a scraping library that increases flexibility.

4.6.2. News aggregation and thematic grouping

Currently, aggregation is implemented at the level of simple article listing, without thematic linking. In this area:

- Automatic topic grouping could be used to connect news stories.
- Duplicate content could be reduced by detecting duplication.

This development would significantly increase the relevance of information and the reader experience.

4.6.3. Development of personalized news feeds

The current system displays a common news feed to all users. However, modern news consumption habits are increasingly moving towards personalized feeds. Future developments include:

- Recording user preferences (e.g. topics, sentiment directions).
- Introducing machine learning-based news recommendation algorithms.

These developments would strengthen the social media-like functions of the system.

4.6.4. Real-time operation

The system is currently updated with a scheduled run. Implementing real-time processing and display of news would further enhance the experience of current events.

5. Summary

The implemented system is overall stable, well-designed, and successfully demonstrates the possibilities of automatic data collection, text processing, and sentiment analysis for Hungarian-language news. During the development process, not only the set functional goals were achieved, but a technological foundation was created that allows for further expansion and refinement of the system. The development opportunities revealed during the evaluation – such as the use of a proprietary sentiment analysis model, thematic grouping of news, the creation of personalized news feeds, or the introduction of real-time data updates – clearly

show that the system has significant development potential. At the same time, these development directions do not reduce the value of the current solution, but rather support its successful implementation and future usability.

References

[1] Dave Winer: *Rss 2.0 at harvard law*. https://cyber.harvard.edu/rss/rss.html

[2] Nottingham, M., Sayre, R. *The Atom Syndication Format*. https://datatracker.ietf.org/doc/html/rfc4287

[3] Richardson, L. *Beautiful Soup Documentation*. https://www.crummy.com/soft ware/BeautifulSoup/bs4/doc/

[4] Arif, A. *What is Puppeteer? Popular Node.js Library Explained*. https://www.webshare.io/academy-article/what-is-puppeteer

[5] Selenium. *Selenium History*. https://www.selenium.dev/history/

[6] Wikipédia. Natural language processing. https://en.wikipedia.org/wiki/ Natural_language_processing

[7] Li, J., Sun, A., Han, J., & Li, C. (2020). A survey on deep learning for named entity recognition. I*EEE transactions on knowledge and data engineering*, 34 (1), 50–70. https://doi.org/10.1109/TKDE.2020.2981314

[8] Wankhade, M., Rao, A. C. S., & Kulkarni, C. (2022). A survey on sentiment analysis methods, applications, and challenges. *Artificial Intelligence Review*, 55 (7), 5731–5780. https://doi.org/10.1007/s10462-022-10144-1

[9] Node.js: https://nodejs.org/en

[10] Node Package Manager: https://nodejs.org/en/learn/getting-started/an-intro duction-to-the-npm-package-manager

[11] Express.js: https://expressjs.com/

[12] Puppeer: https://pptr.dev/

[13] *Huspacy*. https://huspacy.github.io/

[14] *NYTK/sentiment-hts5-xlm-roberta-hungarian*. https://huggingface.co/NYTK/sentiment-hts5-xlm-roberta-hungarian

[15] *XLM-RoBERTa*. https://huggingface.co/docs/transformers/model_doc/xlm-roberta

[16] Ground News: https://ground.news/

[17] *Flask*. https://flask.palletsprojects.com/en/stable/

[18] PostgreSQL: https://www.postgresql.org/