GAZİ
JOURNAL OF ENGINEERING SCIENCES

An Attempt for Price Comparison System

Emre Özakyıldız[a], Oğuzhan Menemencioğlu[*b], Adib Habbal[c]

Submitted: 20.11.2023 Revised: 25.12.2023 Accepted: 27.12.2023 doi:10.30855/gmbd.2023.0705N03

Keywords: Price Comparison Website, Product Matching, Crawler, Scraper, User Centric, Web Application.

[a,] Karabük University, Engineering Faculty, Dept. of Computer Engineering 78050 - Karabük, Türkiye
Orcid: 0009-0004-1568-4465
e mail: emreozakyildiz@gmail.com

[b,*] Karabük University, Engineering Faculty, Dept. of Computer Engineering 78050 - Karabük, Türkiye
Orcid: 0000-0002-4343-6563
e mail: omenemencioglu@karabuk.edu.tr

[c,] Karabük University, Engineering Faculty, Dept. of Computer Engineering 78050 - Karabük, Türkiye
Orcid: 0000-0002-3939-2609

*Corresponding author: omenemencioglu@karabuk.edu.tr

ABSTRACT

In the current landscape, product prices exhibit significant variations across diverse e-commerce platforms. Numerous price comparison websites have emerged to aid users in finding their desired products at the best prices. However, these systems face challenges such as irrelevant search results, outdated product prices, and accuracy issues. This study seeks to address these challenges by developing a user-friendly system. Our system comprises two components. JSoup is employed for data collection, while TF-IDF, SVD, and Cosine Similarity are integrated for product offerings. The proposed platform gathers data from reputable retailers, analyzes it to extract product information, and presents the findings to consumers. We evaluated the system's performance using real data obtained from various marketplaces. The results demonstrate that our system achieved an acceptable accuracy rate when compared to similar industrial solutions and relevant literature. Consequently, users are empowered to make more informed purchasing decisions, leveraging the capabilities provided by the proposed system.

Fiyat Karşılaştırma Sistemi Girişimi

ÖZ

Mevcut ortamda, ürün fiyatları çeşitli e-ticaret platformlarında önemli farklılıklar göstermektedir. Kullanıcıların istedikleri ürünleri en iyi fiyatlarla bulmalarına yardımcı olmak için çok sayıda fiyat karşılaştırma web sitesi ortaya çıkmıştır. Ancak bu sistemler alakasız arama sonuçları, güncel olmayan ürün fiyatları ve doğruluk sorunları gibi zorluklarla karşı karşıyadır. Bu çalışma, kullanıcı dostu bir sistem geliştirerek bu zorlukların üstesinden gelmeyi amaçlamaktadır. Sistemimiz iki bileşenden oluşmaktadır. Veri toplama için JSoup kullanılırken, ürün teklifleri için TF-IDF, SVD ve Kosinüs benzerliği entegre edilmiştir. Önerilen platform, saygın perakendecilerden veri toplamakta, ürün bilgilerini çıkarmak için analiz etmekte ve bulguları tüketicilere sunmaktadır. Sistemin performansını çeşitli pazar yerlerinden elde edilen gerçek verileri kullanarak değerlendirdik. Sonuçlar, sistemimizin benzer endüstriyel çözümler ve ilgili literatürle karşılaştırıldığında kabul edilebilir bir doğruluk oranına ulaştığını göstermektedir. Sonuç olarak, kullanıcılar önerilen sistem tarafından sağlanan yeteneklerden yararlanarak daha bilinçli satın alma kararları verme konusunda güçlendirilmiştir.

Anahtar Kelimeler: Fiyat Karşılaştırma Web Sitesi, Ürün Eşleştirme, Arama Robotu, Veri Toplayıcı, Kullanıcı Merkezli, Web Uygulama

1. Introduction

When it comes to buying products from e-commerce websites, it can be a troublesome for customers to find the cheapest product across various markets. In recent years, many Price Comparison Websites (PCW) have appeared to make this task easier. In Türkiye, there are a few examples of PCW: while Akakce [1] and Cimri [2] compare e-commerce products, Trivago [3] offers the cheapest hotel prices worldwide. Both Akakce and Cimri provide various recommendation options, including unit price sorting, trends and top-rated offerings, to ensure customer satisfaction.

In order to PCW to provide such services, they need to gather up-to-date product data from several sources. The accuracy of product data, covering stock availability, the precision of price information, and exact matching of requested products, is of utmost importance. Additionally, the website's search mechanism holds a major role in its successful operation. Furthermore, tracking the price trends of products over time, allowing users to observe price changes across months, enhances the user experience.

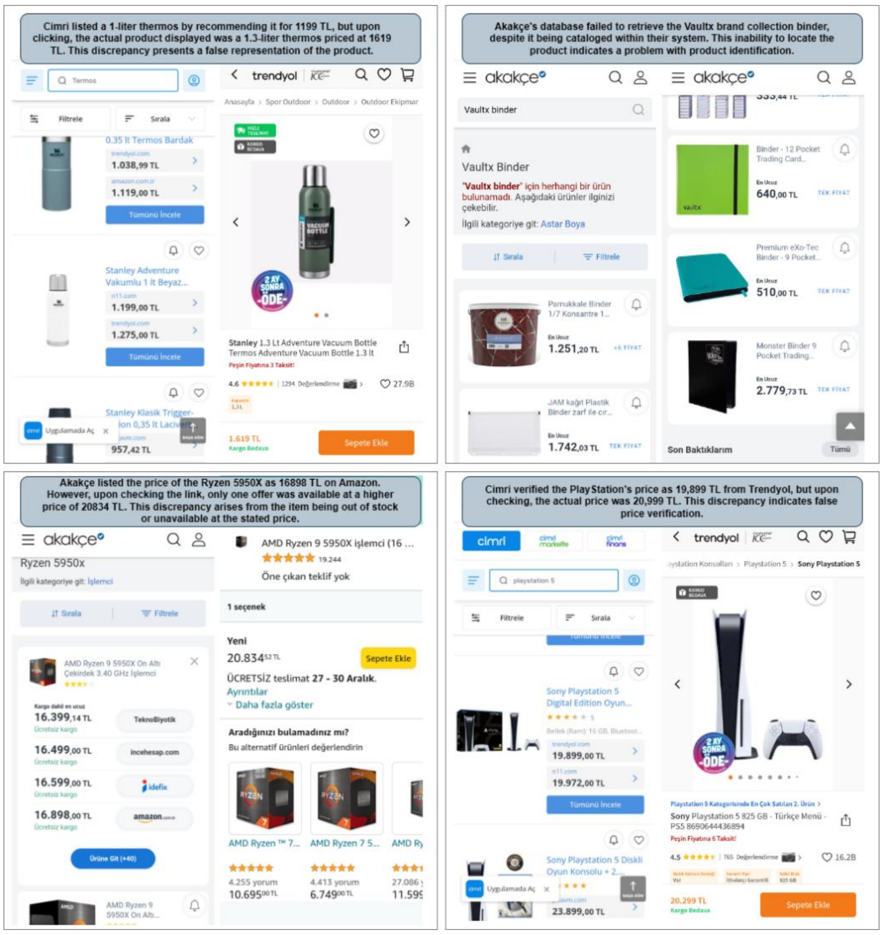
The prominent tools such as web scrappers and web crawlers are used to gather current product information as raw data. In prior studies, diverse approaches have been employed both for web scraping and product matching. For web scraping, researchers such as Nagaraj et al., Shah et al., and Asawa et al. utilized the effective web scraping library BeautifulSoup4 [4-6]. Alternatively, Alam et al. employed the Python library Scrapy [7].

Table 1. Popular Scraper/Crawler Libraries

Library	**Programming Language**	**Type**	**Strengths**	**Weaknesses**
Scrapy	Python	Crawler	+ Full featured web crawling library. + Has detailed crawling abilities like spiders and link extractors.	- limited capability with Javascript (JS)
Apache Nutch	Java	Crawler	+ Comprehensive web crawling framework + Has its own data structure called Apache Hadoop. + Can work with additional related Apache libraries.	- Setting up an Apache Nutch project is complicated when compared with other Java crawlers.
Crawler4J	Java	Crawler	+ Has some of the crawling methods	- Required additional developer written code in demand. - Not up to date.
Puppeteer	Node.js	Scraper	+ JS based scraper so easily can be implemented in JS projects if there's no backend.	- focuses on JS only - there are unofficial ports for Python and PHP
Simple HTML DOM	PHP	Scraper	+ A simple parser for PHP projects. + quick and easy manipulation of HTML elements on small amount of element	- slow when working with large documents or many elements
Beautiful Soup	Python	Scraper	+ HTML/XML Parser library for Python with additional scraping features. + Has its own data structure to manipulate scraped data.	- It doesn't have any crawling ability.
Jaunt/Jauntium	Java	Scraper	+ A simple HTML parser for Java projects. + Jauntium version can handle pages with Javascript using Selenium (actually it is same but built in).	- No crawling features. - Not up to date.
Jsoup	Java	Scraper	+ Has detailed parsing abilities. It uses Document Object Model (DOM).	- Less selector support compared to other scraper libraries. - It doesn't have any crawling abilities.

Prasetyo opted for the Java library Jaunt for the scraping process and utilized the collected data in a K-means algorithm with varying cluster numbers. Notably, the use of ten clusters yielded superior results compared to five clusters [8]. Wang et al. harnessed Jsoup to collect web data, subsequently applying Pearson correlation to the scraped data, although, yielding a weak correlation [9]. In later works, Harikirshnan et al. studied Support Vector Machine (SVM) and compared their performance against Decision Trees and Random Forest algorithms [10]. Additionally, some studies explored alternative data delivery methods for web scraping and product similarity discovery, as demonstrated by Julian et al. [11]. Table 1 includes a comprehensive comparison of the common web scraping and crawler approaches. After analyzing the pros and cons, we conclude that the JSoup is most suitable Scraper library to integrate into our system when we consider the target market and amount of retrieved data.

Existing PCWs, exhibit shortcomings in providing accurate product information to users. Our testing revealed instances where the products offered by the seller market were either unavailable in source PCW or substantially differed from the user's expectations. Furthermore, when a user searches for a certain product, PCWs might list the wrong item, an item with a different price from the original market, or they might list sold-out products, as shown in Figure 1. Therefore, for PCW, there are certain open challenges namely irrelevant search results, outdated product prices, and accuracy issues that should be addressed in future work.


Cimri listed a 1-liter thermos by recommending it for 1199 TL, but upon clicking, the actual product displayed was a 1.3-liter thermos priced at 1619 TL. This discrepancy presents a false representation of the product.

Akakçe's database failed to retrieve the Vaultx brand collection binder, despite it being cataloged within their system. This inability to locate the product indicates a problem with product identification.

Akakçe listed the price of the Ryzen 5950X as 16898 TL on Amazon. However, upon checking the link, only one offer was available at a higher price of 20834 TL. This discrepancy arises from the item being out of stock or unavailable at the stated price.

Cimri verified the PlayStation's price as 19,899 TL from Trendyol, but upon checking, the actual price was 20,999 TL. This discrepancy indicates false price verification.

Figure 1. Issue for current PCWs in tested December 2023

The main goal of this study is to address the prevalent deficiencies observed in existing platforms. These deficiencies encompass inaccurate product availability listings, substantial discrepancies between listed and actual products, and the presentation of unreliable trend prices, all which compromise user confidence and satisfaction. In particular, we aim to develop a user-friendly price comparison system that provides accurate product information, an in-depth product trend analysis and ensures user satisfaction through the provision of precise product offerings.

2. Methods

In the design of the proposed system, there are two phases: Data Collection and Product Offering. Data collection process is handled by Java Spring backend. Meanwhile, the product offering is achieved by using Term Frequency - Inverse Document Frequency (TF-IDF), Singular Value Decomposition (SVD), and cosine similarity. The overall structure of the system from scraping process to product offering level is presented in the following subsections.

2.1. Data Collection

We used Jsoup to scrape products from three popular e-commerce websites: Trendyol (www.trendyol.com), A101 (www.a101.com.tr) and Migros (www.migros.com.tr). The pseudo code of our scraping service is stated in algorithm 1.

JSoup is used to index the categories and subcategories of each market. After the indexing process is completed, scraping is applied to extract products from subcategory pages. Each market has its own CSS selectors for product cards, which is why the scraper code is implemented individually for each market.

Since Migros and Trendyol heavily utilize JavaScript on their websites, JSoup was insufficient for scraping product information from the pages that had been crawled because such markets load price information by using frontend programming, which makes it hard to get price data from the website without running JavaScript code before the scrapping process. JSoup does not have a built-in web driver to run JS code. In that scenario, Selenium was used to execute JavaScript code on the page sources, and JSoup was employed to retrieve the page content afterward.

Algorithm 1: Web Scraping Pseudo Code

```
for each market in markets do
    for each category in market.categories do
        for each sub-category in category.subCategories do
            page <- JSoup.connect(market)
            products <- page.select(market.productCardCssSelectror)
            for each product in sub-category.products do
                new Product();
                Product.setProductName(product.select(marketSpecificNameSelector));
                Product.setProductLink(product. select(marketSpecificLinkSelector));
                Product.setProductPrice(product. select(marketSpecificPriceSelector));
                Product.setProductImage(product. select(marketSpecificImgSelector));
                Product.setSubCategory(sub-category);
                Product.setTimestamp(currentTimestamp);
                Product.save(Products);
            end for
        end for
    end for
end for
```

The Data Access Layer utilizes Java Persistence API (JPA) to efficiently store product information within the PostgreSQL database. This integration ensures effective data management and retrieval. Figure 2 is the database diagram that provides a visual representation of the project's data structure, illustrating the relationships and organization of the stored information.

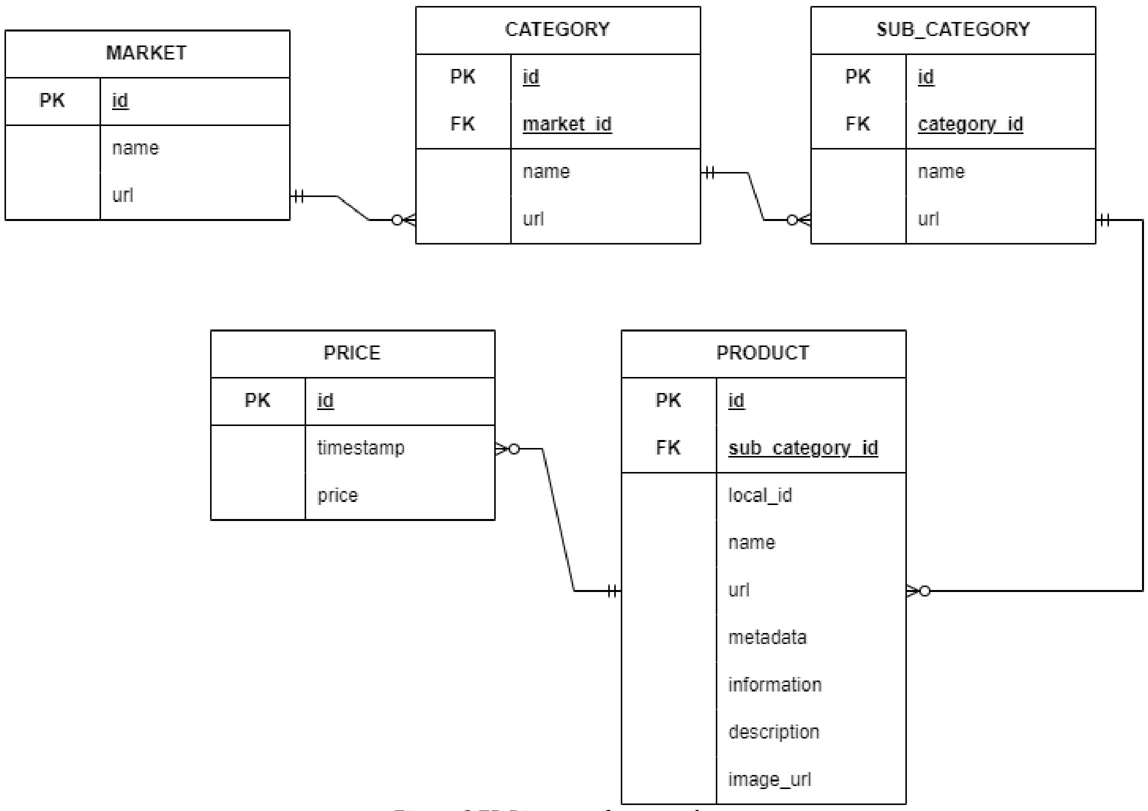


Figure. 2 ER Diagram of proposed system

2.2. Product Offering

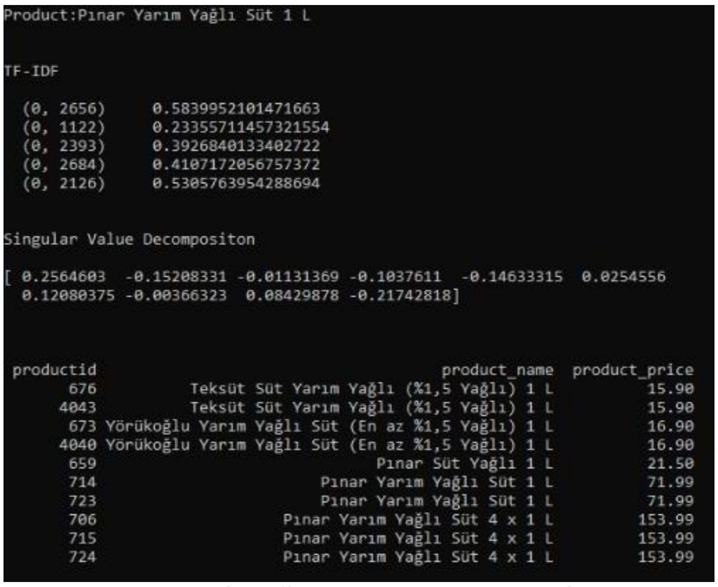
The offering phase initiates with a product request submitted by the user. The requested product ID is subsequently forwarded to the Python backend, which in turn retrieves the top 10 similar items associated with the requested product.

Algorithm 2: Recommendation Pseudo Code

```
1    TFIDF = fit(product_data_with_features)
2    svd= SVD.fit(TFIDF,10)
3    similarities = cosine_similarity(svd)
4    sorted = sort(similarities)
5    for i, product in enumerate(products):
6        similarProducts = GetTopSimilarIndices(sorted, 20)
7        top10[product] = similarProducts
8    return top10
```

The product offering described in Algorithm 2, utilizes, TF-IDF and SVD [12]. TF-IDF is employed to convert product information, including product names, prices, and categories, into numerical representations. The obtained numerical representations are then used to calculate similarities and identify the top 10 most similar products in the database by SVD and cosine similarity [13].

To compute similarities, the Cosine Similarity Metric [14] is favored for its proven effectiveness in handling high-dimensional textual data. Its inherent capability to evaluate similarities while adjusting for variations in scale within product descriptions establishes it as an efficient measure for product matching tasks. The proficiency of this metric in identifying similarities based on orientation rather than magnitude in numerical representations aligns seamlessly with the diverse nature of product attributes.



```
Product:Pınar Yarım Yağlı Süt 1 L


TF-IDF

  (0, 2656)     0.5839952101471663
  (0, 1122)     0.23355711457321554
  (0, 2393)     0.3926840133402722
  (0, 2684)     0.4107172056757372
  (0, 2126)     0.5305763954288694


Singular Value Decompositon

[ 0.2564603  -0.15208331 -0.01131369 -0.1037611  -0.14633315  0.0254556
  0.12080375 -0.00366323  0.08429878 -0.21742818]



 productid                                   product_name  product_price
       676          Teksüt Süt Yarım Yağlı (%1,5 Yağlı) 1 L          15.90
      4043          Teksüt Süt Yarım Yağlı (%1,5 Yağlı) 1 L          15.90
       673 Yörükoğlu Yarım Yağlı Süt (En az %1,5 Yağlı) 1 L          16.90
      4040 Yörükoğlu Yarım Yağlı Süt (En az %1,5 Yağlı) 1 L          16.90
       659                              Pınar Süt Yağlı 1 L          21.50
       714                        Pınar Yarım Yağlı Süt 1 L          71.99
       723                        Pınar Yarım Yağlı Süt 1 L          71.99
       706                    Pınar Yarım Yağlı Süt 4 x 1 L         153.99
       715                    Pınar Yarım Yağlı Süt 4 x 1 L         153.99
       724                    Pınar Yarım Yağlı Süt 4 x 1 L         153.99
```

Figure 3. Product offering process using TF-IDF and SVD

An example of this process is illustrated in Figure 3 with real data. To retrieve best match, TF – IDF calculates numerical metrics for “Pınar Yarım Yağlı 1 Lt Süt” milk, then SVD weights the most important part of the product and cosine finds the similarities between the grabbed products within the system. We prefer only 10 matches to test. It is worth noting that increasing the pool size may lead to improved results in specific cases. However, in general, it can result in irrelevant product offerings due to the variations of product description.

2.3. System Architecture

The architecture is illustrated in Figure 4. The proposed system incorporates two distinct backend systems. In the initial step, Java is responsible for gathering product data from various markets and storing this data in the database. This process iterates until all market data has been collected, ensuring the continuous updating of product information.

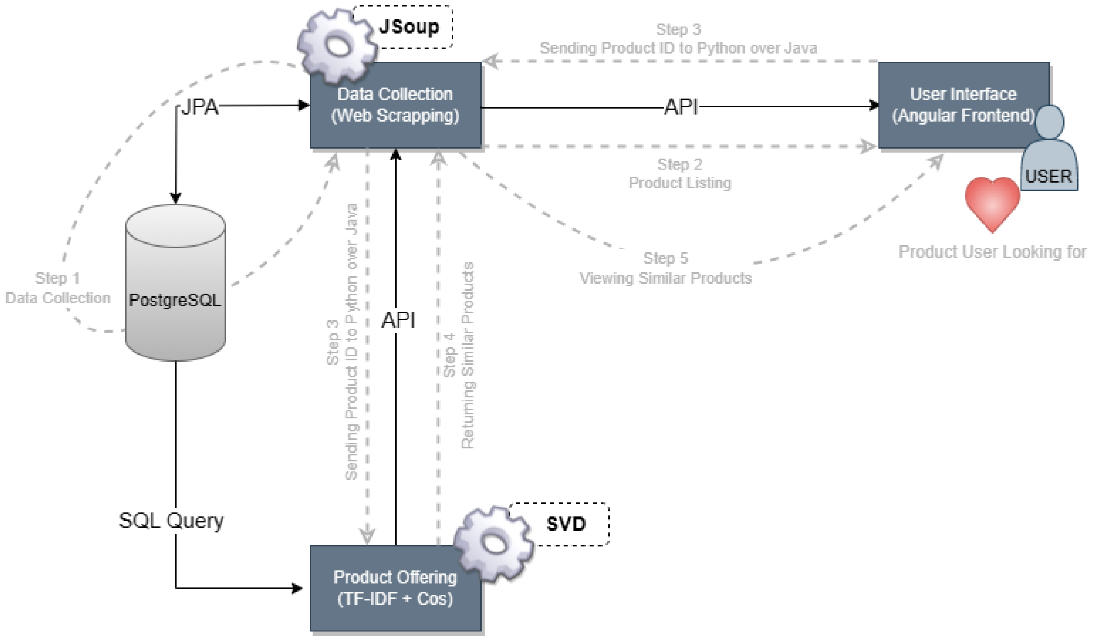

JSoup
Step 3
Sending Product ID to Python over Java
JPA
Data Collection (Web Scrapping)
API
User Interface (Angular Frontend)
USER
Step 2
Product Listing
Step 1
Data Collection
PostgreSQL
Step 3
Sending Product ID to Python over Java
API
Step 4
Returning Similar Products
Step 5
Viewing Similar Products
Product User Looking for
SQL Query
SVD
Product Offering (TF-IDF + Cos)

Figure. 4 Proposed system architecture

Meanwhile, users engage with the front end to search for a product. In the second step, products retrieved by the backend are presented to the user. In the third step, the selected product's identifier is transmitted to Python via a Java API.

Python, in the fourth step, Python is used to develop our data offering algorithm to recommend products that are similar to the previously selected item. The identifiers of these generated products are returned to Java. Subsequently, in the fifth step, the product information for these similar products is gathered and then relayed to the frontend. So, a user finds intended products list.

The front-end portal showcased in Figure 5 represents the result of the user search. Our system offers users access to both price trends and the identification of the most cost-effective products available.

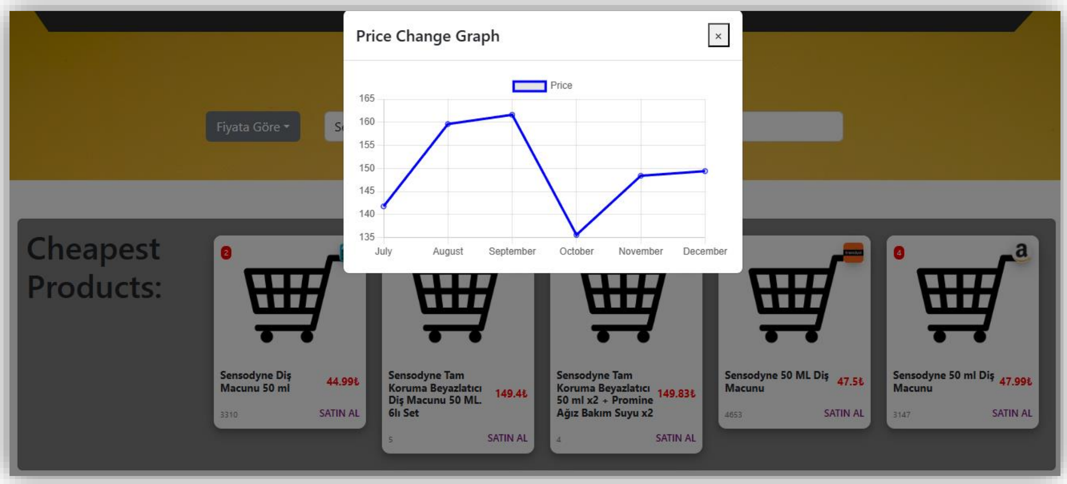

Price Change Graph
×
Price
165
160
155
150
145
140
135
July
August
September
October
November
December
Fiyata Göre
Cheapest Products:
Sensodyne Diş Macunu 50 ml
44.99₺
3310
SATIN AL
Sensodyne Tam Koruma Beyazlatıcı Diş Macunu 50 ML 6lı Set
149.4₺
5
SATIN AL
Sensodyne Tam Koruma Beyazlatıcı 50 ml x2 + Promine Ağız Bakım Suyu x2
149.83₺
4
SATIN AL
Sensodyne 50 ML Diş Macunu
47.5₺
4653
SATIN AL
Sensodyne 50 ml Diş Macunu
47.99₺
3147
SATIN AL

Figure 5. User interface

3. Results and Discussion

The proposed system was evaluated through a comparative performance analysis with two of Türkiye's most popular PCWs, namely Akakçe and Cimri. For testing purposes, we selected Amazon's top-selling products list from September 18th to September 24th and conducted individual product searches on each of the websites.

For our evaluation, distinct performance metrics were measured to gauge the effectiveness of our system compared to Akakçe and Cimri. These metrics encompassed several key aspects:

- the "first place match," denoting when a product recommended by the platform appeared as the primary suggestion;
- "match," indicating any instance of product alignment across search result pages;
- "outdated product," evaluating discrepancies in product information such as stock availability or pricing accuracy; and
- "missing product," signifying products untraceable on the platform.

Figure 6 illustrates that the product matching algorithms employed by both Akakçe and Cimri yield very close matching ratios, in contrast to our system. This outcome was derived from human testing, involving the manual search for selected products on each comparison website. While Akakçe exhibits the highest first-place product matching rate, exceeding 75 percent, all systems maintain a nearly 90 percent matching rate.

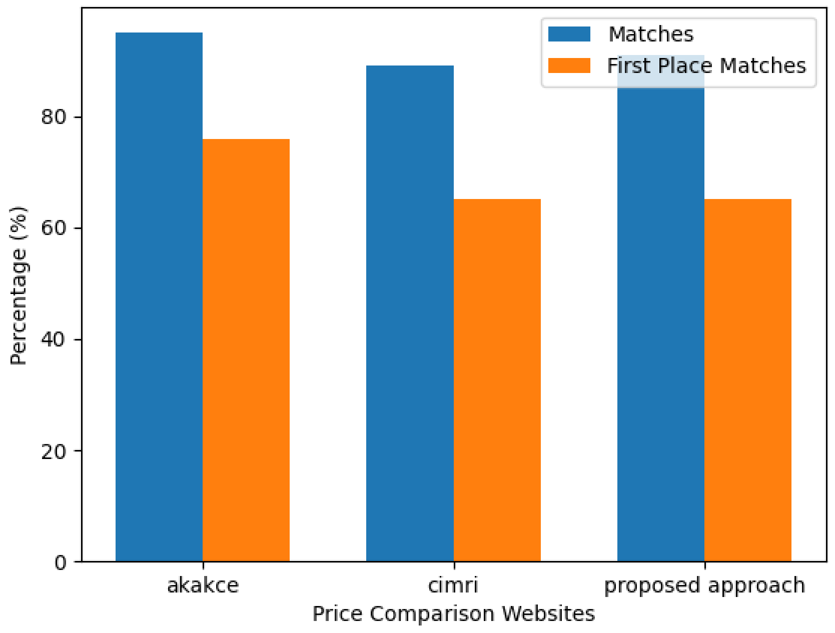


Figure 6. Matching Rates

Table 2 has success rates of previous studies. Those rates stand for the system's overall performance. Success rate in this study is the match rate of the searched products.

Table 2. Success rate comparison of previous studies

Author	**Success Rate (%)**
Ketki Gupte et al. [10]	68.34
Vincentius Riandaru Prasetyo [5]	73.80
Evan Shieh et al. [12]	83.00
Zhixiang Lin et al. [11]	90.12
Harikirshnan K. Et al. [7]	93.00
Proposed approach	84.00

Upon examining the timeliness and accuracy of product data from previously matched products (as shown in Figure 7), our system consistently provides the most recent and accurate product information, minimizing false directives. However, we should note that our system has data from only three markets while others have over ten suppliers.

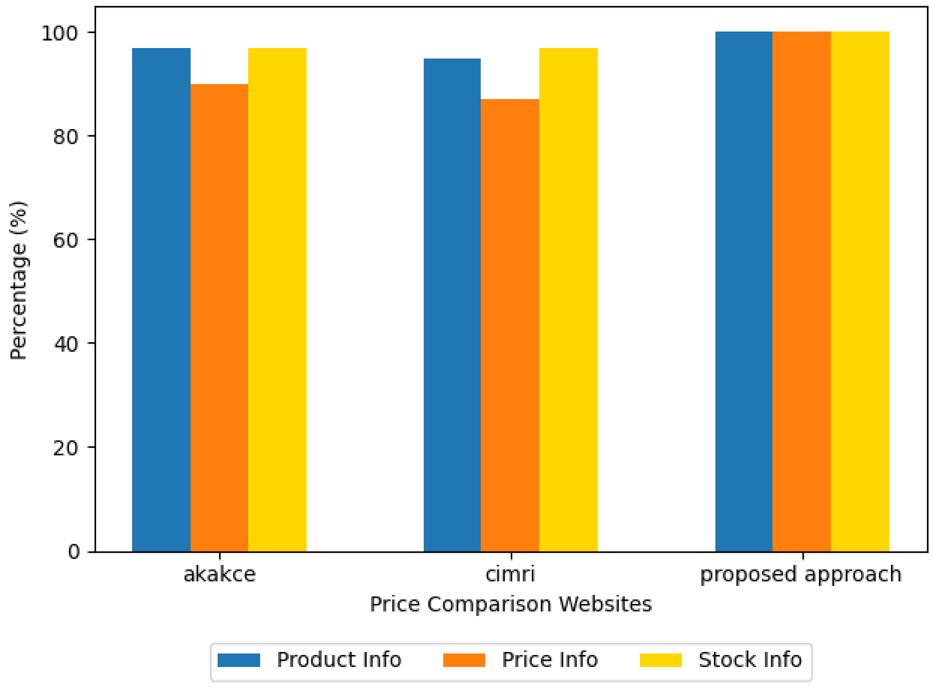


Figure 7. Data Accuracy

To address mismatched products, we further investigated whether these items were included in the respective comparison websites or were absent from their databases. Figure 8 reveals that 17.5 percent of the mismatched items in Cimri were present in their system but remained undiscoverable by users.

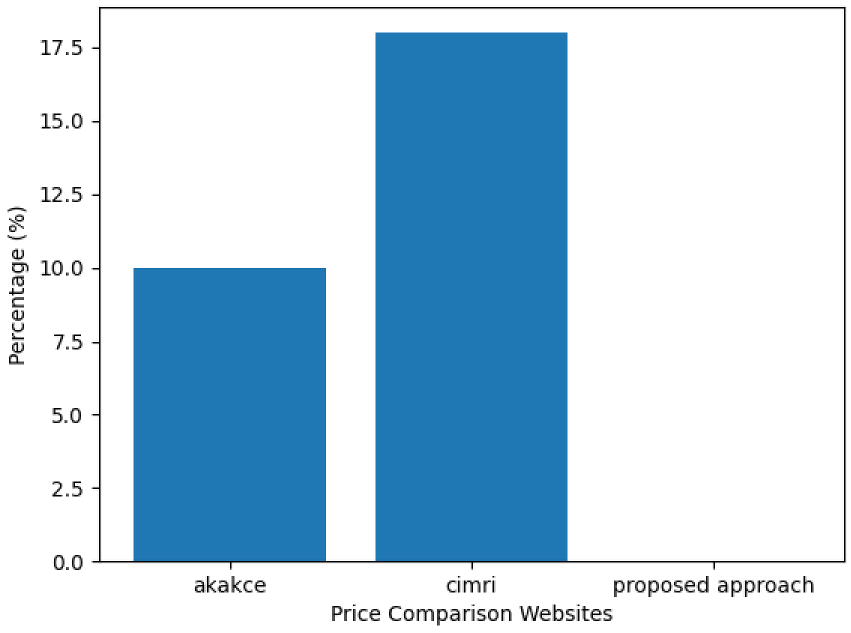


Figure 8. Outdated Data Results

Finally, Figure 9 presents an assessment of the system's efficiency which is calculated by comparing the price shown on PCW and on the actual website of the market, specifically its ability to identify the most cost-effective products. As depicted, our system's performance falls between that of Akakçe and Cimri.

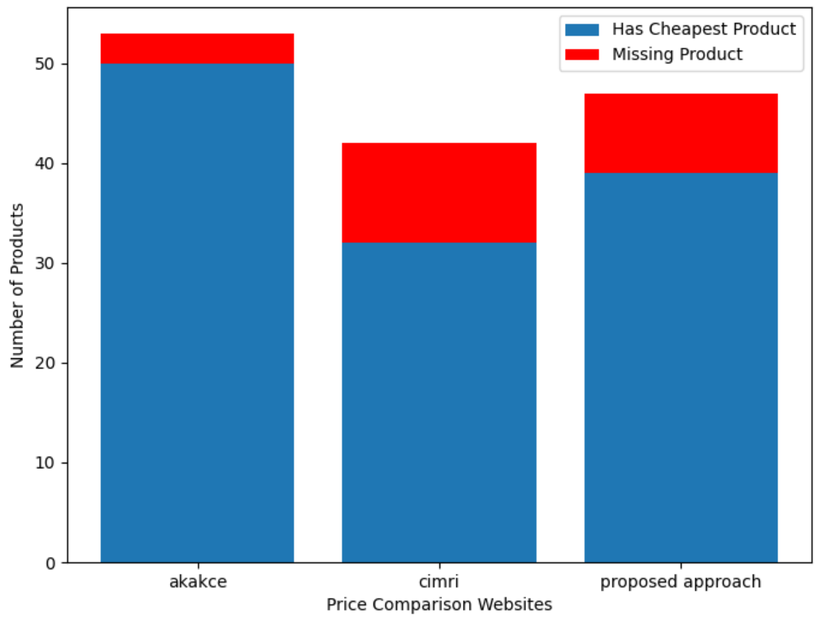


Figure 9. Cheapest Product Rate

While the results affirm the competitiveness of the proposed approach against its counterparts, further research may lead to clarify its strengths and identify areas for enhancement. The initial objective entails an expanded product database and market inclusion, prioritizing user-centric objectives. This strategic expansion aims to ensure a comprehensive product availability, thereby enhancing the overall user experience within the PCW. Another crucial facet of future works involves augmenting the product matching algorithms. The envisaged increase in market diversity, and subsequently in product volume, may poses a challenge to the current SVD modelling. As a response, alternative algorithms, such as Random Forest, will be explored to ascertain their viability and comparative performance against the existing models.

4. Conclusion

This study introduced an alternative price comparison system. Our proposed system consists of two main components, data collection and product offering. The former deploys Jsoup scraper library and the later integrates TF – IDF, SVD, and cosine similarity. The proposed system was evaluated, and its performance compared with existing academic research and industrial PCWs. When it is compared with literature, the success rate reached 84% percentage which is reasonable. Furthermore, our system achieved acceptable efficiency and accurate product offering compared to Akakçe and Cimri.

While our system demonstrates promising results in delivering current and accurate product information, it indicates opportunities for enhancing its performance from different aspects. Future research endeavors aim to verify the system's strengths by diversifying the market, including more products, and exploring alternative matching algorithms, such as Random Forest. These strategies will be pivotal in enhancing the system's competitiveness and user-centric functionality within the price comparison landscape.

Acknowledgement

This paper improves Emre Özakyıldız and Fatma Birel B.Sc. thesis research at Karabuk University in 2023, supervised by Dr. Oğuzhan Menemencioğlu. Fatma Birel's work was separately supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) under TÜBİTAK 2209-A Science Fellowships and Grant Programmes and the Project ID is: 1919B012218690.

Conflict of Interest Statement

The authors declare that there is no conflict of interest

References

[1] “Akakçe.” [Online]. https://www.akakce.com. [Accessed Dec. 23, 2023].

[2] “Cimri.” [Online]. https://www.cimri.com. [Accessed Dec. 23, 2023].

[3] “Trivago.” [Online]. https://www.trivago.com.tr. [Accessed Dec. 23, 2023].

[4]P. Nagaraj, V. Muneeswaran, A. V. S. R. Pavan Naidu, N. Shanmukh, P. V. Kumar, and G. S. Satyanarayana, “Automated E-Commerce Price Comparison Website using PHP, XAMPP, MongoDB, Django, and Web Scrapping,” in *2023 Int. Conf. Comput. Commun. Informatics, ICCCI 2023*, no. Iccci, pp. 1–6, 2023, doi:10.1109/ICCCI56745.2023.10128573

[5] R. Shah, K. Pathan, A. Masurkar, S. Rewatkar, and N. Vengurlekar, “Comparison of E-commerce Products using web mining,” *Int. J. Sci. Res. Publ.*, vol. 6, no. 5, p. 640, 2016.

[6] A. Asawa, S. Dabre, S. Rahise, M. Bansode, K. T. Talele, and P. Chimurkar, “Co-Mart - A Daily Necessity Price Comparison Application,” in *Proc. - Int. Conf. Appl. Artif. Intell. Comput. ICAAIC 2022*, no. Icaaic, pp. 1076–1080, 2022, doi:10.1109/ICAAIC53929.2022.9792935

[7] A. Alam, A. A. Anjum, F. S. Tasin, M. R. Reyad, S. A. Sinthee, and N. Hossain, “Upoma: A Dynamic Online Price Comparison Tool for Bangladeshi E-commerce Websites,” in *2020 IEEE Reg. 10 Symp. TENSYMP 2020*, no. June, pp. 194–197, 2020, doi:10.1109/TENSYMP50017.2020.9230862.

[8] V. R. Prasetyo, “Searching Cheapest Product on Three Different E-Commerce Using K-Means Algorithm*,” in Proceeding - 2018 Int. Semin. Intell. Technol. Its Appl. ISITIA 2018*, pp. 239–244, 2018, doi:10.1109/ISITIA.2018.8711043.

[9] J. Wang, S. Yang, Y. Wang, and C. Han, “The crawling and analysis of agricultural products big data based on Jsoup,” in *2015 12th Int. Conf. Fuzzy Syst. Knowl. Discov. FSKD 2015*, pp. 1197–1202, 2016, doi:10.1109/FSKD.2015.7382112.

[10] K. Harikirshnan., R. Nagavigneshwar., R. Vignesh., R. Santhosh, and R. Reshma, “Intelligent Online Shopping using ML-based Product Comparison Engine,” in 2023 International Conference on Inventive Computation Technologies (ICICT), Apr. 2023, no. Icict, pp. 174–179, doi:10.1109/ICICT57646.2023.10134401

[11] L. R. Julian and F. Natalia, “The use of web scraping in computer parts and assembly price comparison,” in 2015 3rd International Conference on New Media (CONMEDIA), Nov. 2015, pp. 1–6, doi:10.1109/CONMEDIA.2015.7449152

[12]S. Brunton and J. Nathan Kutz, “Singular Value Decomposition (SVD),” in Data-Driven Science and Engineering, no. Dmd, Cambridge University Press, 2019, pp. 3–46.

[13] J. Han, M. Kamber, and J. Pei, “Getting to Know Your Data,” in Data Min., pp. 39–82, Jan. 2012, doi:10.1016/B978-0-12-381479-1.00002-2

[14]A. R. Lahitani, A. E. Permanasari, and N. A. Setiawan, “Cosine similarity to determine similarity measure: Study case in online essay assessment,” in Proc. 2016 4th Int. Conf. Cyber IT Serv. Manag. CITSM 2016, Sep. 2016, doi:10.1109/CITSM.2016.7577578.