zyte

ZYTEPAPER

Comparative analysis and evaluation of the quality of web product data extraction

A study of data quality achieved with commercial extraction services and open source libraries

By Konstantin Lopukhin,
Zyte Data Science
Team Lead

Introduction

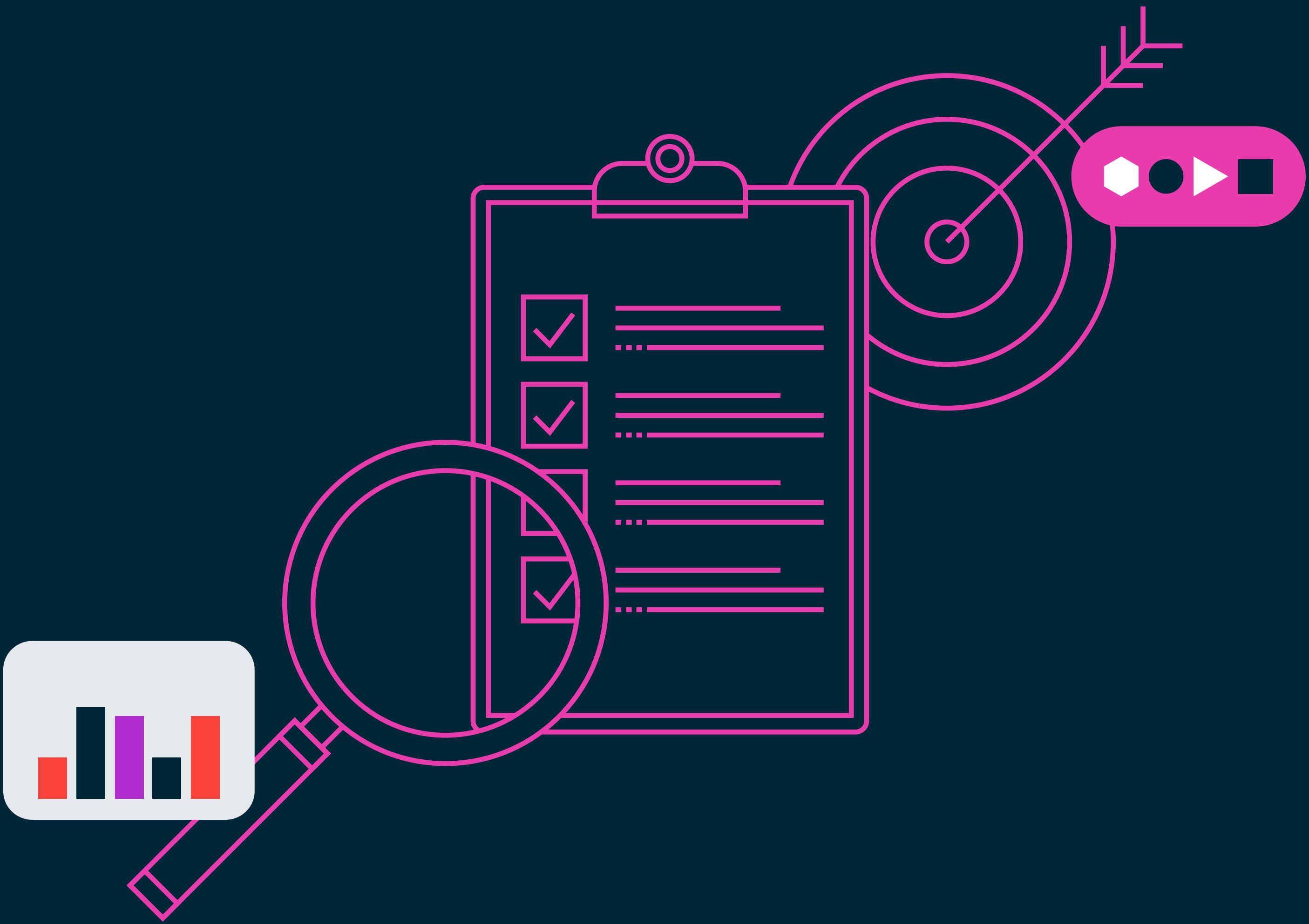
This paper describes an evaluation of the quality of product extraction using Zyte's Automatic Extraction Product API, measured against comparable results achieved by Diffbot, another commercial extraction service, and an open-source baseline we developed.

We present the goals of the project and discuss how the dataset was collected. We also describe procedures used to ensure the accuracy and trustworthiness of the product data extraction process.

The results of the evaluation are offered together with conclusions, observations and suggestions.

This study follows a similar investigation conducted by Zyte in 2020, where we measured the quality of article body extraction achieved by our own Automatic Extraction API against other commercial and open source alternatives.

Read our 2020 paper on article extraction.



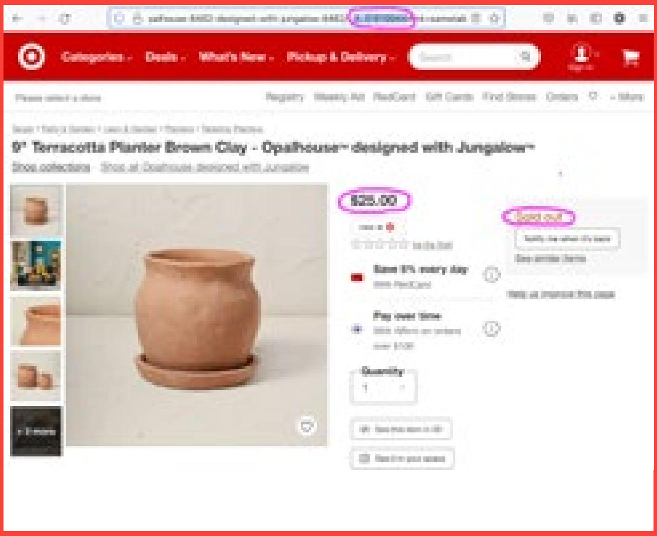
What is product extraction?

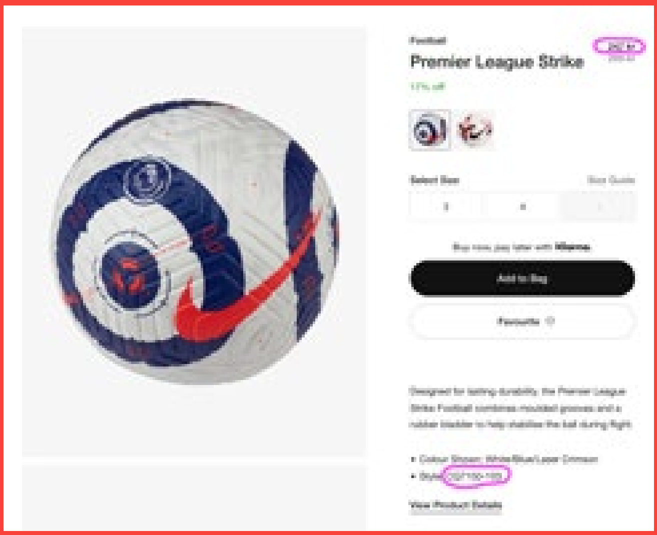
The goal of product extraction is to obtain the key attributes of a particular product, starting with the URL of a webpage featuring a single main product.

Within the context of this exercise we regard a 'product' to be virtually any type of consumer good. Real estate (property), vehicles and events were treated as out of scope.

In this study we have not evaluated the task of finding the product on the website or crawling, assuming that the product URL is already given.

Furthemore we have not evaluated page download quality, considering only cases where a page was successfully downloaded.





Above are two screenshots of product web pages, highlighting the attributes - price, SKU and availability (in/out of stock) - that were considered in this study.

Comparing product and article extraction

Product extraction is typically more challenging than the relatively simpler task of extracting relevant content from the body of a news article or blog.

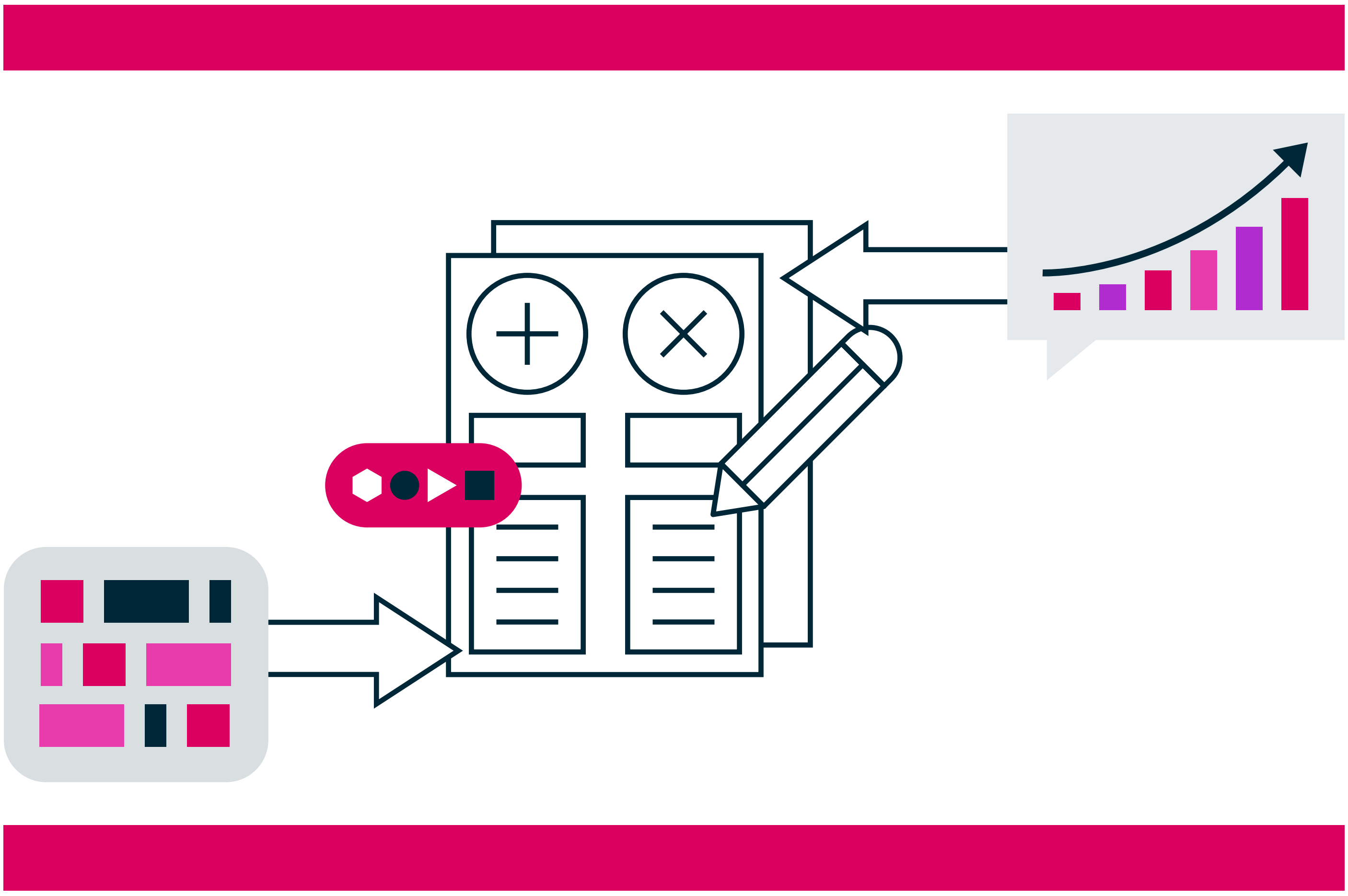
This is because products have a lot more attributes, while the websites themselves vary significantly in terms of their appearance and how they work under the hood.

Another consideration is that a product page often contains not only the main product, but other related products as well.

Moreover, many attributes can be missing sometimes, even the price, and the extraction system must not pick a price from another product on the same page.

Furthermore, product-focused sites tend to feature heavier use of dynamic content - such as javascript - than simpler text-based article pages.

Watch our webinar that discusses our evaluation of the performance of alternative solutions for article body extraction, and the results achieved by our Automatic Extraction API.



Methodology

The principal goal of our investigation was to compare the quality of extraction results achieved with Zyte's AI-powered Automatic Extraction Product API against a commercial competitor.

The extraction quality of the following systems was therefore evaluated:

Zyte Automatic Extraction Product API

(also referenced as 'Zyte' in this document). Pages were fetched using version 2021.2.0.

Diffbot Product API

(also referenced as 'Diffbot' in this document). Pages were fetched during February 2021.

As a baseline open-source extractor we used extruct, a widely-used open source tool that extracts embedded metadata from HTML markup, and price-parser (also referenced as 'extruct' in this document). This extractor uses semantic markup: as such its recall is far from perfect.

Against this reference point we compared our own Automatic Extraction Product API and Diffbot, an established commercial service that has already established a high bar for extraction quality. Like our own solution, Diffbot features an API that allows individual URLs to be submitted.

By feeding each of these solutions with a carefully selected set of real-world product page URLs, our objective was to evaluate which approach yielded the best quality results in terms of extracting these key product attributes:

- **Price**
- **Availability** (whether a product is in-stock or out-of-stock)
- **SKU** (Stock Keeping Unit)

Note that pages were served as snapshots from a server we control, so the domain was the same for all sites, and different from the original web-site domain. This means that if some systems had custom per-domain extractors, they wouldn't be active, so the extraction quality might have been worse due to this.

Expressing data quality

As the main metric we chose per-attribute **F1**, a measure of data quality that combines **recall** and **precision**. F1 is widely used as an objective measurement of extraction quality, as it accommodates cases of commonly occurring attributes as well as rare ones.

In this context, the recall of a particular attribute - such as product price or SKU - is the ratio of correctly identified attribute values relative to the total number of attribute values in the dataset.

Precision, meanwhile, expresses the ratio of correct predictions for some attribute, relative to the total number of predictions for this attribute.

An F1 score is thus expressed as the harmonic mean of recall and precision. This is a useful way of averaging two numbers which is higher when they are better balanced, while heavily penalising extremely low precision or recall.

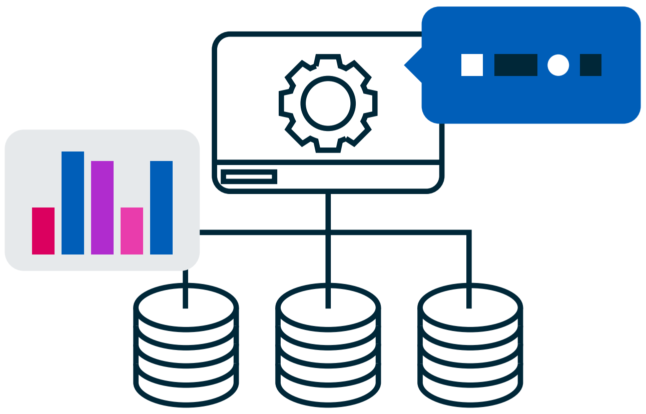
Here's an illustration. If a system gave a correct price prediction on four pages and an incorrect prediction on one page, while in the dataset price is annotated on ten pages, then recall would be computed as 0.4 (4 divided by 10), while precision would be 0.8 (4 divided by 4 plus 1). The F1 value in this case would be approximately 0.53 (or 2 times 0.8 times 0.4 divided by the sum of 0.8 and 0.4).

Collecting the dataset

To make the experiment as fair as possible, we took considerable pains to minimise factors that could undermine the credibility of the test and the results.

Rather than 'cherry-picking' web domains that may have unfairly skewed the results in our favour, we asked two extraction experts outside the Zyte data science team to propose an unbiased set of popular consumer product domains.

Their selections ranged from Tier-1 marketplaces including Amazon, Ebay and Alibaba to well-known mono-brand sites such as Ikea and John Lewis. These were complemented by some sites from more obscure brands and vendors. From these, we selected a broad spectrum of URLs including front page products, more deeply hidden items, discounted and out-of-stock products.



We also took other precautions, like taking a 'snapshot' of our chosen target URLs before feeding them into each extraction engine. That way we could be sure that page content had not altered in any way in the short intervals between each test run, and was consistent irrespective of the download location.

URLs were collected in two stages:

Popular product domains, with seven URLs collected per site:

- **Amazon** (four URLs were collected from each of the following regions: US, UK, DE, IT, FR)
- **Walmart**
- **Target**
- **Home Depot**
- **Ebay**
- **Zara**
- **John Lewis**
- **H&M**
- **Alibaba**
- **Rakuten**
- **Zalando**
- **Best Buy**
- **Canadian Tire**
- **Ikea**
- **Wayfair**

Less popular domains:

We consulted two site catalogues and sampled 50 sites from each catalogue, collecting two positive URLs from each site.

When selecting URLs for this experiment, our goal was to ensure variety. We wanted products from the main page, plus some more deeply hidden products, with and without discounts, and including some out-of-stock products.

Serving snapshots

On many websites product prices and availability can vary depending on factors such as the visitor's location and time of day. To ensure that all systems receive the same input, we captured the snapshots of the pages and served them using pywb. The snapshots included both the main page, and all sub-resources required for rendering, such as images, JS, CSS, etc. pywb is a web archiving capture and replay framework for python, implementing the basic functionality of a 'Wayback Machine'.

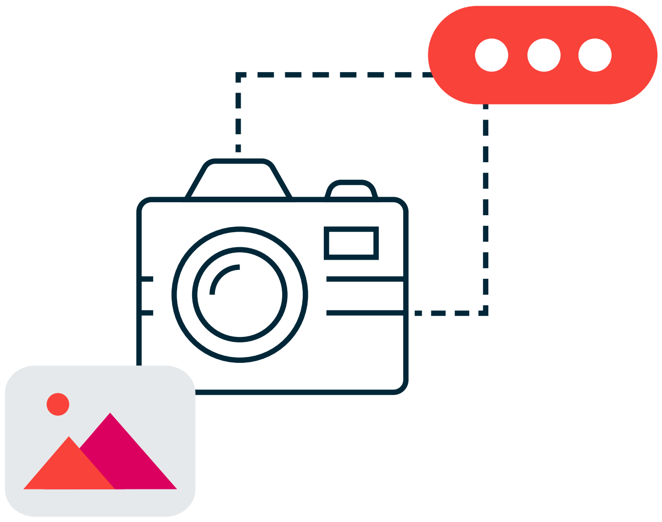
Some websites were still broken even when serving from a snapshot, either because they tried to obtain fresh product details, or due to rendering being different from what was seen in the browser. Such pages were excluded from the dataset.

Snapshots can be prepared with the following command, assuming the dataset-warc folder contains the WARC snapshots (available as an archive from the releases section of the repo: dataset-warc.zip):

```
docker run --rm -it \
  -e INIT_COLLECTION=product-
extraction-benchmark \
  -v `pwd`/dataset-warc:/dataset-
warc \
  -v `pwd`/pywb-data:/webarchive \
  webrecorder/pywb:2.5.0 \
  wb-manager add product-
extraction-benchmark /dataset-
warc/*.warc.gz
```

Then we can do extra configuration to remove the frame:

```
docker run --rm -it \
  -v `pwd`/pywb-data:/webarchive \
  --entrypoint /bin/bash \
  webrecorder/pywb:2.5.0 \
  -c "echo ‚framed_replay: false' > /
webarchive/config.yaml && touch /
webarchive/templates/banner.html"
```



And then serve the snapshots:

```
docker run --rm -it -p 80:8080 \
  -v `pwd`/pywb-data:/webarchive \
  webrecorder/pywb:2.5.0 \
  wayback -a
```

You can use **dataset/pywb-mapping.json** to map between page IDs and URLs served by pywb.

Dataset

The dataset is included in our published repo in several forms

html files (gzip-compressed)

are included directly in the repo under dataset/html with file names corresponding to page IDs used in **dataset/ground-truth.json**

Screenshots of pages (before snapshot creation)

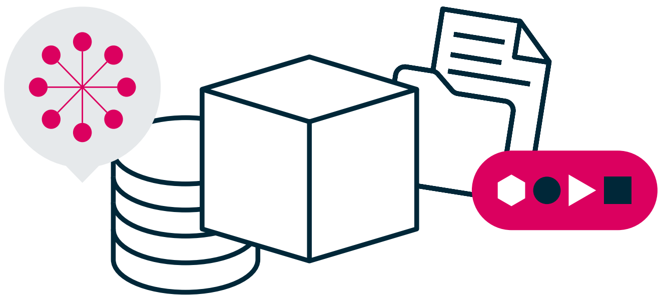
are included as an archive in the releases section of the repo: dataset-jpeg.zip, file names corresponds to page ids used in **dataset/ground-truth.json**

Snapshots in WARC format

which can be served by pywb are included as an archive in the releases section of the repo: dataset-warc.zip, file names have the form <**DATETIME>-<ID>.warc.gz**, where **<ID>** corresponds to page ids used in **dataset/ground-truth.json**

Annotations are available under **dataset/ground-truth.json**, which looks like this:

```
{
  "0094967f37c647407d92b6242027
36e3272231cf61f8ccaeca0b19aeb63
1ab28": {
    ...
    "brand": [
      “365 by Whole Foods Market”
    ],
    "gtin": [
      “099482467999”
    ],
    "sku": [
      "B07FW264WL"
    ],
    "url": "https://www.amazon.com/
dp/B07FW264WL"
  },
  "01efbb747904557e90dcb56d1f255
a4034d5206c796539e2ce4d9dd6867
2c098": {
    ...
```



Each evaluated attribute can have multiple values in the ground truth: all values are considered correct.

Evaluating the data

Evaluation was performed by the **evaluate.py** script, which requires Python 3.6+ and **tabulate** dependency.

The main metric is F1. Each attribute can have multiple ground truth values, but at most one predicted value is allowed. Predictions from all systems are available under **dataset/output.**

Here are some attribute-specific notes:

Price was matched as a decimal number, and currency was not evaluated.

Availability can have one of two possible values: **InStock** and **OutOfStock**. This is a required attribute: in case it's not clear that a product is out-of-stock, it is assumed to be in-stock. For systems which can have an empty value (Zyte, extruct), we filled empty values with **InStock** to match Diffbot.

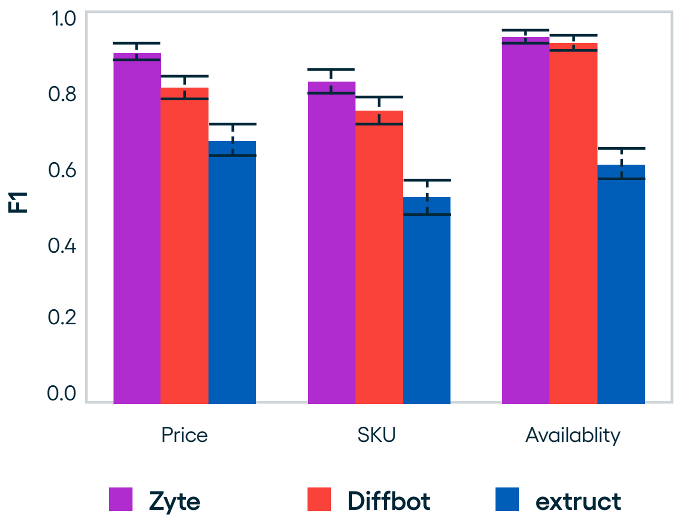
SKU is a Stock Keeping Unit. It is quite flexible, and can be any identifier which is used by the website to uniquely identify the product. Based on this, we used the following guidelines:

- Even if a SKU was explicitly spelled out on a website, but the system extracted a different SKU from another place (e.g. Diffbot often extracts a SKU from the URL), we considered it to be correct if it looked like a unique identifier.
- If such an identifier was also a GTIN (Global Trade Item Number), it was also considered correct since it is globally unique.
- If such an identifier was a part of the product name (like an MPN), then this was not considered a correct SKU since it's not guaranteed to be unique across the website (unless it would have been an MPN in a mono-brand shop; however there were no such cases in this exercise).
- If an extracted SKU had extra text in it (e.g. product code 123 instead of 123), then this was considered not correct. While it would likely still serve as unique, it may happen that the prefix would not always be present or might be different.
- Non-critical differences in SKU were allowed, e.g. 123/123 instead of 123123, or 123 instead of P123, as long as it was clear it does not break uniqueness. For example ‚PU_106160/01' vs ‚106160' was not allowed.

Results

Based on the product page dataset as described, we found extraction quality achieved by Zyte Automatic Extraction to be significantly better than Diffbot in terms of price and SKU attributes. The quality of results for availability are comparable between Zyte and Diffbot.

The main results of the evaluation are presented in the chart below. We used the F1 metric for each attribute: this is a measurement that combines the precision and recall of the model. The higher the F1 score, the better quality of extraction: hence a 'perfect' model has an F1 score of 1. We also show standard deviation in black. This indicates how reliable F1 values are, and how they might change in a different dataset.



Detailed results of the comparison are presented in the table below. In addition to F1 we present precision and recall values, with standard deviation given after the "±" sign.

Also shown is Support for each attribute, which is the number of pages where an attribute should be present according to ground truth annotations. We also present the metrics for InStock and OutOfStock values of the product availability attribute.

We see that for both price and SKU attributes, precision and recall of Zyte Automatic Extraction is significantly higher compared to Diffbot, while both solutions performed above the baseline. We also see that the difference in recall is greater than the difference in precision - this means that Diffbot falls short in finding the expected value, although the quality of the value it finds is also lower.

The difference in availability is not significant as the dataset is somewhat biased towards products that are in stock: there are only nine out-of-stock products, so availability score is dominated by products which are in-stock, while out-of-stock score is uncertain.

Extruct returned significantly inferior results for all attributes. In the table below we see that this is primarily caused by poor recall, while precision is often competitive, for example for price it has higher precision compared to Diffbot. This means that when it does return an extraction, it is often correct, but it misses a lot of valid cases. This happens because extruct obtains results from semantic markup, which is not always present, but if it is present, it is usually correct.

Attribute	System	F1	Precision	Recall	Support
Price	Zyte	0.918 ± 0.023	0.918 ± 0.024	0.918 ± 0.024	134
	Diffbot	0.824 ± 0.031	0.844 ± 0.032	0.806 ± 0.034	134
	extruct	0.685 ± 0.039	0.864 ± 0.036	0.567 ± 0.044	134
SKU	Zyte	0.841 ± 0.031	0.860 ± 0.030	0.822 ± 0.033	135
	Diffbot	0.765 ± 0.035	0.828 ± 0.035	0.711 ± 0.039	135
	extruct	0.537 ± 0.045	0.786 ± 0.049	0.407 ± 0.043	135
Availability	Zyte	0.957 ± 0.018	0.957 ± 0.018	0.957 ± 0.018	140
	Diffbot	0.943 ± 0.020	0.943 ± 0.020	0.943 ± 0.020	140
	extruct	0.626 ± 0.041	0.905 ± 0.034	0.479 ± 0.043	140
InStock	Zyte	0.977 ± 0.010	0.970 ± 0.015	0.985 ± 0.010	131
	Diffbot	0.970 ± 0.011	0.956 ± 0.018	0.985 ± 0.011	131
	extruct	0.954 ± 0.014	0.954 ± 0.019	0.954 ± 0.018	131
OutOfStock	Zyte	0.625 ± 0.154	0.714 ± 0.184	0.556 ± 0.174	9
	Diffbot	0.429 ± 0.175	0.600 ± 0.254	0.333 ± 0.166	9
	extruct	0.333 ± 0.147	0.333 ± 0.168	0.333 ± 0.167	9

Sources of error

Here we present a qualitative analysis of errors in specific attributes only for Zyte and Diffbot solutions. extruct achieves significantly lower extraction quality, and most of its errors can be attributed to poor recall.

Price

Diffbot:

Often extracts the price from any number close to the product, such as number of reviews, SKU or other variables. We also observe that recall suffers more than precision.

Zyte:

Many errors come from semantic markup indicating a price that differs from what's shown on the page.

For example, a price with discount is shown but semantic markup shows the regular price.

Other errors come from bad price parsing, such as '34.000' being parsed as '34000'. In some cases the price in the main product is empty, but the system is erroneously picking up a price from a related product, while it should be producing an empty price instead.

Availability

Diffbot:

The most common error is failure to detect that an item is OutOfStock (unavailable).

Zyte:

Similarly, the most common error is failure to detect that an item is OutOfStock. In a few cases, an incorrect OutOfStock prediction is made when some sizes are still InStock.

SKU

Diffbot:

Many missing SKUs (Stock Keeping Units). Sometimes picks up an MPN (Manufacturer Part Number) as an SKU. Is able to extract SKUs from URLs. Also we observe that recall suffers more than precision.

Zyte:

Sometimes erroneously picks up an MPN as an SKU. Unable to extract SKUs from URLs.

Conclusion and observations

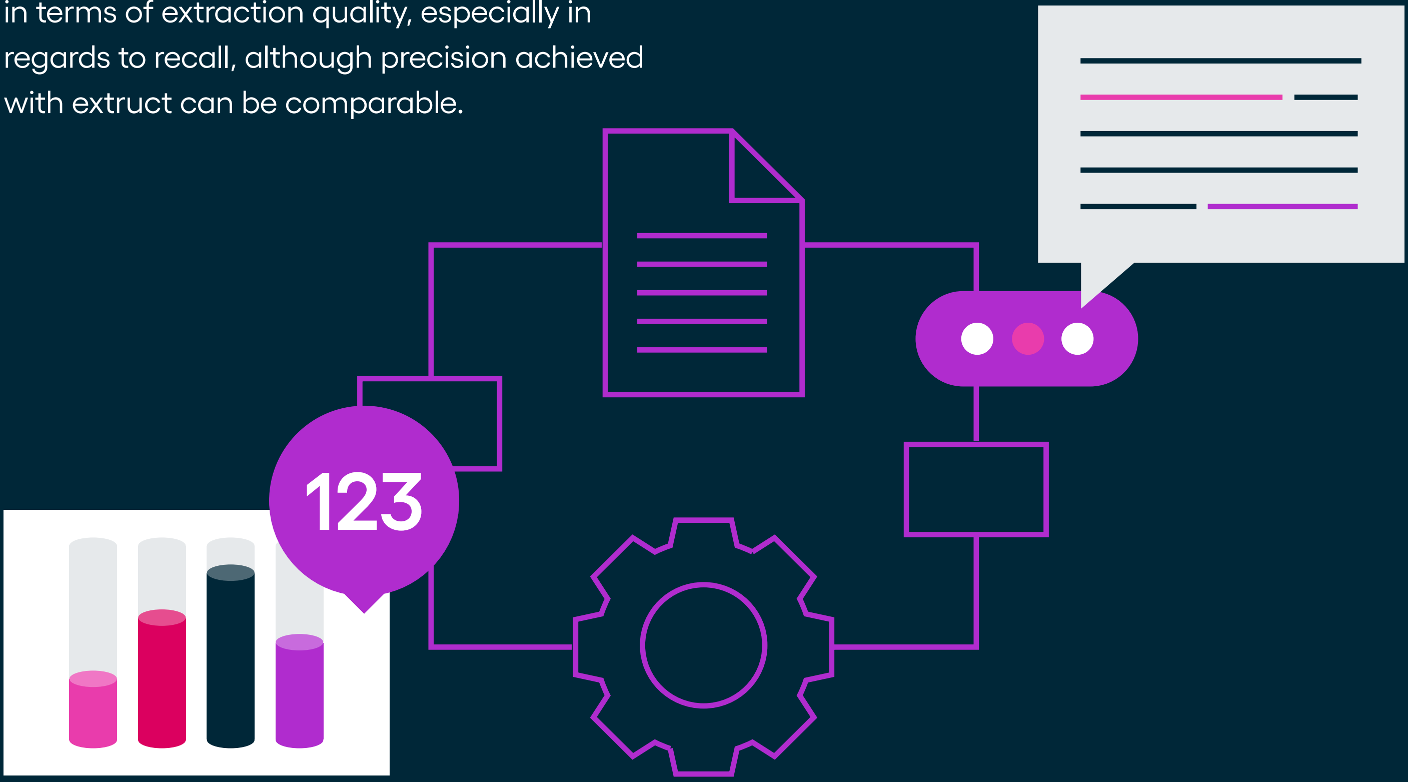
Overall, we were pleased to find that Zyte achieved significantly better extraction quality than Diffbot for price and SKU attributes. Both solutions achieved similar quality for availability.

Nevertheless, we found a number of potential areas for future improvement in the performance of Zyte's product extraction algorithm - for example with regards to SKU extraction from URLs and price parsing issues. We also discovered that it takes considerable effort to ensure a fair evaluation when website content varies for different visitors due to issues such as geographical location or time of day. Noting these challenges, we remained satisfied with the 'snapshots' approach we had chosen.

In contrast with article extraction, we also confirmed our suspicion that open source solutions lag behind commercial services in terms of extraction quality, especially in regards to recall, although precision achieved with extruct can be comparable.

What's more, commercial solutions can handle downloading: this is a significant task in itself, especially when crawling at scale. The open source solution we have studied here takes HTML as input, which is already downloaded. Downloading HTML is trivial in simple cases: however product websites often ban automated downloaders, making the task more difficult.

We also realised that the evaluation methodology itself can be improved. It would be great to cover more attributes, such as currency and brand, and expand the size of the dataset. Availability evaluation suffered from a small number of products that were out of stock - it would be desirable to balance them out with in-stock items.



Open for inspection

We have released the entire dataset for the study described in this paper on Github.

This resource includes web archive files, test methodology, screenshots of chosen pages, ground truth annotations, evaluation code and baseline open source extraction code.

Download the dataset

zyte

See for yourself

Want to find out how Zyte can help you get high quality, timely product extraction data at scale?

Sign up and try our Automatic Extraction product API for free.

At Zyte we turn websites into data with industry leading technology and services.

Our solutions include:

- **Data Extraction Service**
  Let our web scraping experts build and manage the bespoke data extraction solution for your business needs.
- **Automatic Extraction powered by AI**
  Instantly access accurate web data through our user-friendly interface or various Extraction APIs and save time getting the data you need.
- **Smart Proxy Manager (formerly Crawlera)**
  Forget about proxy lists. We manage hundreds of thousands of proxies, so you don't have to.
- **Data extraction platform**
  Access developer tools, data extraction APIs and documentation, built and maintained by our world-leading team of over 100 extraction experts.

zyte

It's yours. The web data you need.

Access clean, valuable data with web scraping services that drive your business forward.

Talk to us

www.zyte.com

Cuil Greine House, Ballincollig Commercial Park Link Road, Ballincollig, Co. Cork, Ireland