Using Active Mediators and Passive Extractors Inside Materialized Data Integration Systems

**Paulo V. M. Cardoso, Flavio F. Franzin,
Sergio L. S. Mergen**

[1]Curso de Ciência da Computação – Universidade Federal de Santa Maria (UFSM)
Santa Maria – RS – Brazil

{pcardoso, ffranzin, mergen}@inf.ufsm.br

Abstract. *Materialized data integration architectures are generally composed by active extractors and passive mediators, where the extractors have the goal of forwarding to the mediator all identified objects that may be of interest. This one hand communication road may lead to the difficulty of identifying relevant objects (specially if the source is not well structured) and a waste of energy analyzing irrelevant objects and possibly extracting them. Also, there is a processing overhead associated with the deduplication of objects that are already mapped. In this paper we propose an extension of the integration architecture that allows mediators to play a more active role, guiding passive extractors as to what objects need to be extracted and how they can be identified. We present a case study that shows how passive and active extractors can coexist under the same data integration system.*

1. Introduction

Materialized integration systems are built on top of architectures that allow generally three things: extracting data from the sources, mapping the extracted data into a unified (global) schema and finally accessing the materialized data from the unified schema to answer queries[Anter et al. 2016].

The extractors (in their many forms) have the goal of finding objects. The identified objects are send to a component called mediator. If more than one data source publishes overlapping information about an object, the mediator is responsible for merging the data so that each object is represented only once in the unified schema. What is worth noting here is that, during the extraction phase, there is typically a one way communication between the source and the mediator, where the source forwards all (possibly) relevant information about objects it was able to find. In other words, the extractors play active roles while the mediator is entirely passive. There are some caveats to this approach:

- If the source is semi-structured or unstructured, it may be hard to identify objects within the contents of a data source. For instance, suppose the system integrates information about artists and the supported extractor collects artists names from wikipedia articles. There are many named entities inside texts written in natural language. It is cumbersome for a extractor to locate a value inside an article and to conclude it is indeed not only a person's name but the name of an artist.

- The extractor may visit many sources that are not relevant in the context of the integration system. In the best case scenario, there is a waste of energy processing such sources. In the worst case scenario, it may lead to the mapping of irrelevant objects. For instance, the extractor of artists name may (unnecessarily) harvest the whole wikipedia to find what it was looking for.
- The system needs to check if an incoming object already exists in the unified schema, so that the two versions are merged into one. This matching process can be very time-consuming if all mapped objects need to be verified.

There are many proposals aimed at handling these issues: the extraction of named entities is by itself a research topic inside the natural language processing area. Focused crawlers are studied as a way to restrict the search to relevant sources only. Object matching and string matching are handful techniques to allow fast and precise object deduplication. However, considering these ideas are based on heuristics, there is always a chance of failure.

To minimize this sort of problems, in this paper we propose an extension of the general extraction architecture of a materialized data integration system by introducing the concepts of active mediators and passive extractors. Instead of having a mediator that merely receives information from the sources, it is given the freedom to define which object to look for, based on the objects already stored in the repository. Once the extractor receives that information, it proceeded with the extraction of information that complement the existing object. The complemented object is then send back to the mediator.

This paper is organized as follows: Section 2 shows how the extension can be accommodated in a general architecture for data extraction. It also shows how the active and passive extractors can coexists and how one can benefit from the other. Section 3 presents a case study regarding extracting data into a biological collaborative system. Both active and passive extractors are implemented. Section 4 presents the results obtained when executing the case study. Section 5 discusses related work. Finally, Section 6 brings our concluding remarks.

2. Supporting Active Mediators and Passive Extractors

Figure 1 shows a general data extraction architecture modified to support active mediators and passive extractors. We only care about materialized data integration systems, where data from the sources are fed into a global repository. The solid rectangle surrounds the traditional parts whereas the dotted rectangle surrounds the proposed extension.

An extractor is labeled as active if it receives no information from the mediator about the objects to extract. Its purpose is to identify all information it deems to be relevant. The information is than shaped in the form of objects represented in a standardized canonical format. Once the passive mediator receives the collected objects, it needs to store them. Also, the mediator needs to check if the object already exists in the repository. If so, it needs to merge the overlapping objects into one.

On the other hand, an extractor labeled as passive receives information from the mediator about the object to extract. The information (a message) can be anything that helps the extractor locate relevant data sources or parts of a data source that contain relevant information. When the extractor is passive, its purpose is to identify information

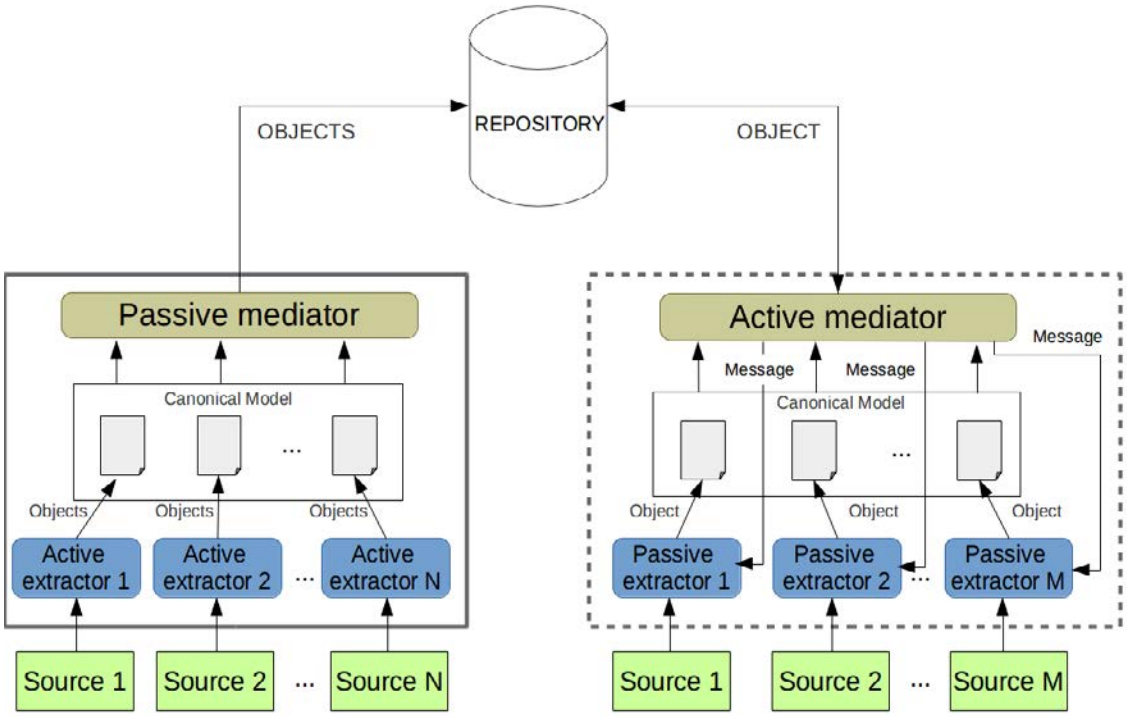

Figure 1. Data Extraction Architecture: distinction between active/passive extractors and active/passive mediators

about a single object, the one the mediator is asking for. The extracted object is described in the canonical format and send back to the mediator. Once the active mediator receives the collected object, it needs to store it in the repository, merging the preexisting object with the collected one.

Algorithm 2.1 shows the interaction between the active mediator and the passive extractor. The mediator prepares a message for each object in the repository. The extractor receives the message and use it as a guide to locate data sources and find the desired object.

EXTRACT(repository)

1: **for each object** O from the repository **do**
2: $message \leftarrow$ extract_message(O)
3: $O_{new} \leftarrow$ run_passive_extractor($message$)
4: merge(O,O_{new})
5: **end for**

Algorithm 2.1: COMMUNICATION BETWEEN AN ACTIVE MEDIATOR AND A PASSIVE EXTRACTOR

Active and passive extractors can live together inside the same system. Active extractors can be used to create an initial representation of an object. Once the object exists in the global repository, passive extractors can be triggered to complement the object.

Passive extractors lack the ability to find new objects. However, they are more suited if the purpose is to enrich an existing object with new information, since it is more

capable to spot the parts of the source that describe the object, specially if the data source is not well structured. Besides, since a passive extractor knows what to look for, it is able to focus on data sources that are indeed relevant, instead of striving across unfitting sources. Finally, active mediators have no need to check if the object exists in the repository. It already knows the object exists, and keeps the older version ready to merge once the extractor finishes collecting data.

It is important to notice that we are not defining a particular format (or content) of the message, and neither a particular merging strategy. Our intention here is to define the foundations of the architecture and leave the details to real implementations. In the next section we discuss a case study where this architecture is implemented.

3. Case Study: BioID

BioID is a collaborative system developed by the students that authored this paper. The system is targeted at species labeling and classification. Generally speaking, it allows users to index life forms and their properties, such as habitat, location and ranks (e.g. species, genus, family, class, kingdom). The properties are expansive, so users can enrich the description of an object by informing additional properties.

Figure 2 shows a snapshot of the system where a species is wrapped inside a block. The block contains information such as common name, images, properties and comments. The user is allows to add new information, but cannot edit information registered by other users. An overview of the main features of BioID is presented in [Cardoso et al. 2015].

Figure 2. BioId's block with a life form information

According to [Mora et al. 2011], there are approximately 8,7 million species on the planet, and only 1,2 million that has a taxonomic classification. Also, there are large digital repositories that publishes information about classified species, such as *Cites*[1] and *CalPhotos*[2]. However, none of them allows collaborative working.

[1](https://www.cites.org/
[2](http://calphotos.berkeley.edu/

Given that a lot of information is already available online, we decided to create active and passive extractors to populate the BioID repository. The purpose of the extractors is to locate objects concerning life forms (animals, to be precise) and represent each one of them in a canonical format supported by the BioID system. The format of an object is as described in Definition 1.

Definition 1 *(Object) An object O in the accepted canonical format is a tuple $\{P, I\}$. Element P is a list of properties $\{p_1, p_2, p_3, ..., p_n\}$, where $p_i = \{key, value\}$. Element I is a list of images $\{i_1, i_2, i_3, ..., i_j\}$, where $i_j = \{uri\}$. The key, $value$ and uri elements are literals.*

The key and $value$ pair refers to the name of a property and its corresponding value, respectively. The uri refers to the location where a image was found. Also, we use the term 'incoming object' to refer to an extracted object that is not yet processed by the mediator. Conversely, the term 'stored object' is used to refer to an object already stored in the repository.

Two active and three passive extractors were implemented by the students who authored this paper. One of the active extractors collects objects from *Cites*. This data source publishes a CSV file with rank properties of catalogued animals. The second active extractor collects objects from *CalPhotos*, a source that publishes a CSV file with species common names. Since the sources are structured, it was relatively straightforward to create the extractors.

All of the passive extractors process HTML data from the following web pages: *Google Images*, *CalPhotos* and *Wikipedia*. The first two were given the task to complement objects with images. On the other hand, the *Wikipedia* variant needed to complement an object with images and properties related to ranks, according to the biological classification of a life form. For instance, 'species' and 'genus' are possible ranks to extract.

Recall that passive extractors need a message to guide the extraction. Given a stored object O in the repository, and its list of properties $P \in O$, the message is created as the value of a property $p \in P$, such that $p.key$ = 'species'. If the object has no property whose key is species, the message is not created and the extraction for the given object is aborted. For instance, possible values for the message are 'H. sapiens' and 'E. coli'. The reasoning behind this heuristic is that the species property serves to uniquely identify a life form. Therefore, its value is useful for searching additional information about an object.

Once the message is generated, the extractors use it as described in the following steps:

1. A *url* is accessed. The *url* is formed by a fixed part and a dynamic part formed by the message. For instance, the Portuguese wikipedia extractor uses `https://pt.wikipedia.org/wiki/` as the fixed part.
2. The part of the web page where relevant data can be found is detected. For *Google Images* and *CalPhotos*, the relevant parts (images) are those that appear enclosed by the img tag. For *Wikipedia*, the relevant parts (properties and images) can be found inside a div tag named *infobox*.
3. The relevant information is described as an object in the canonical format. When multiple images exist inside the web page, only the first one is extracted (the first

img tag).
4. The retrieved object is merged with the stored one. Next section explains how the merging is performed.

3.1. Identification and Merging of Duplicated Objects

Definition 2 presents a notion of duplicated properties and images, which is important for the identification and merging of duplicated objects.

Definition 2 *(Duplicated Property and Duplicated Image) Two properties p_i and p_j are duplicated if $p_i.key = p_j.key$. Similarly, two images i_i and i_j are duplicated if $i_i.uri = i_j.uri$.*

Duplicated objects are naturally identified within active mediators, as Algorithm 2.1 showed. On the other hand, passive mediators need to the check if an incoming object O_i has a duplicated object O_s in the repository, where $P_i \in O_i$ and $P_s \in O_s$. We do this by checking if properties $p_i \in P_i$ and $p_s \in P_s$ are duplicated, having 'species' as key and the exact same value. In other words, objects of the same species are inferred to be duplicated.

The merging of two duplicated objects O_i (incoming object) and O_s (stored object) follows the same reasoning for active and passive mediators, as Table 1 describes:

image $i_i \in O_i$ **and image** $i_s \in O_s$	**property** $p_i \in O_i$ **and property** $p_s \in O_s$
If i is not duplicated in O_s • i_i is added to O_s.	If p_i is not duplicated in O_s • p_i is added to O_s.
If i_s and i_i are duplicated • no action is performed.	If p_s and p_i are duplicated • $p_s.value = \text{concat}(p_s.value, ' , ', p_i.value)$

Table 1. Rules for Merging Properties and Images of Duplicated Objects

We observe that this strategy is clearly limited, as we do not take into account the similarity of elements. For instance, the properties with keys $color$ and $colour$ would be considered unlike. Also, the above rules may end up concatenating equal values (e.g. $color = red, red$). We are aware of the limitations, that we leave as future work. We remark that our intention in this paper is to demonstrate how active and passive mediators can be used in combination, regardless of their particular weaknesses.

4. Experimental Results

This Section presents how effective the proposed extraction methods are for finding relevant objects (in the case of the active extractors) and finding complementary information about objects (in the case of passive extractors).

Figure 3 shows the number of animals found by the active extractors along with the relative proportion with respect to the number of species that have a biological classification. As shown, the extractors found 25.820 objects from the two visited data sources. After the object deduplication and merging, 24.680 distinct animals were indexed. This is approximately 1,43% of the total of classified species. If only animals are considered (the actual target of the extractors), the percentage grows to 2,58%.

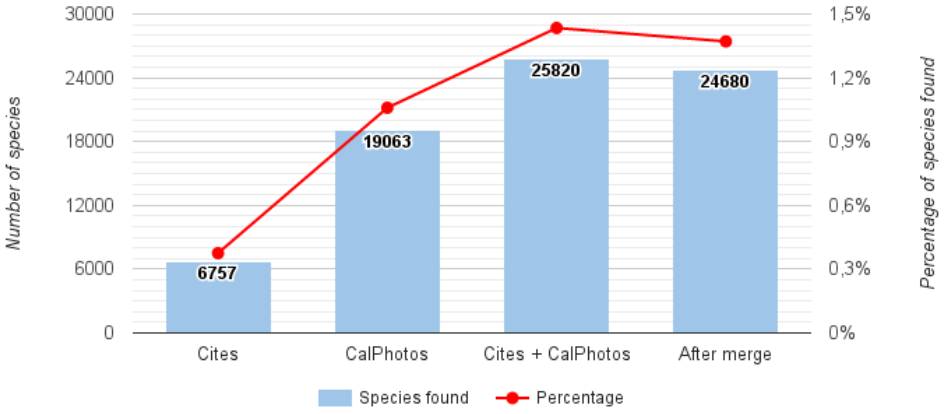

Figure 3. Animals found by the implemented active extractors.

Figure 4 shows how the extractors enrich objects with rank properties. The total amount of properties was computed as the product of the number of indexed objects and the seven rank properties (Kingdom, phylum, class, order, family, genus and species). The percentage value is the ratio between the amount of properties found and the total.

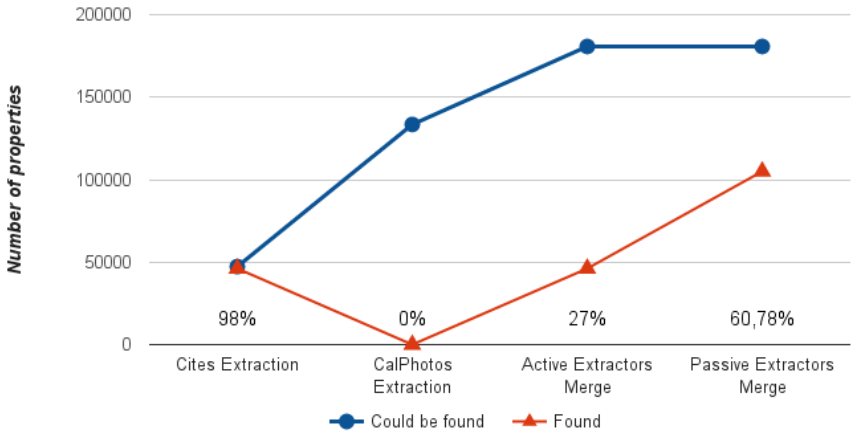

Figure 4. Number or rank properties found by active and passive extractors

The *CalPhotos* active extractor was unable to find the rank properties (only the common name was found). Conversely, the *Cites* active extractor found 46.310 properties out of 47.299, reaching 6,83 properties per animal (a 98% recall). After the deduplication and merge, we reached an average of 1,87 properties per animal (a 27% recall). After merging the indexed objects with the results coming from the *Wikipedia* passive extractor, we reached an average of 4,25 properties per animal (a 60,78% recall).

Another interesting usage of the passive extractors was to complement the indexed objects with images related to the corresponding animal. Table 2 shows the achieved

results. As we can see, *Google Images* was able to find images for almost every indexed animal, whereas wikipedia had images for approximately half the objects. Summing up, the average number of images per animal is greater than two. Only 47 animals remained without images.

Table 2. Images found by the passive extractors

Data source	Found Images	Images per animal
Google Images	24644	0,999
CalPhotos	18952	0,768
Wikipedia	13439	0,545
Total	57035	2,311

To conclude, we evaluate how reliable are the images found by the passive extractors. To help in the evaluation, we developed a tool that exhibits images for randomly chosen indexed animals and lets users indicate if the image really correspond to the animal (hit or not hit). The more hits obtained, the more precise are the passive extractors. Four settings were used, with fifty animals each, considering images extracted from i)*Google Images*, ii)*CalPhotos*, iii)*Wikipedia* and iv) all of them together. Two specialists participated of this experiment (one professor and one PhD student from areas related to biological research).

The results are presented in Figure 5. The hit and not hit values are an average of the number of hits and not hits from the two participants, respectively. As we can see, most of the sampled images are rightfully linked to an animal. Precision varies from 86% (*CalPhotos*) to 96% (*Google Images*). The extraction of the non corresponding images were due to the usage of unreliable data sources (e.g. *Google Images*). In other cases, the incorrect extraction was a symptom of the passive extractors deficiencies. The correction of the extraction algorithm (and possibly the adjustment of the message send by the mediator) is left as future work.

5. Related Work

The idea of helping extractors do their job is not really new. There are many possible ways of leveraging extraction methods. Here we discuss a few of them, namely focus crawling, wrapper induction and user-assisted extraction.

Focus crawlers are aimed at finding relevant sources (generally web pages) based on previously crawled relevant sources or a list of seeds provided by a user. There are several alternatives to select which pages to visit. Recent proposals achieve this by a diversity of machine learning algorithms that classify the sources as relevant or irrelevant[Safran et al. 2012, Dong and Hussain 2014].

Wrapper induction is a research area where the goal is to automatically create (or adjust) wrappers so they are able to extract new information from data sources. Previous knowledge is important to achieve this task. For instance, [Senellart et al. 2008] proposes a way to automatically create wrappers for the hidden web (data that sits behind HTML forms). An important part of the work is the extraction of data from the result pages. This

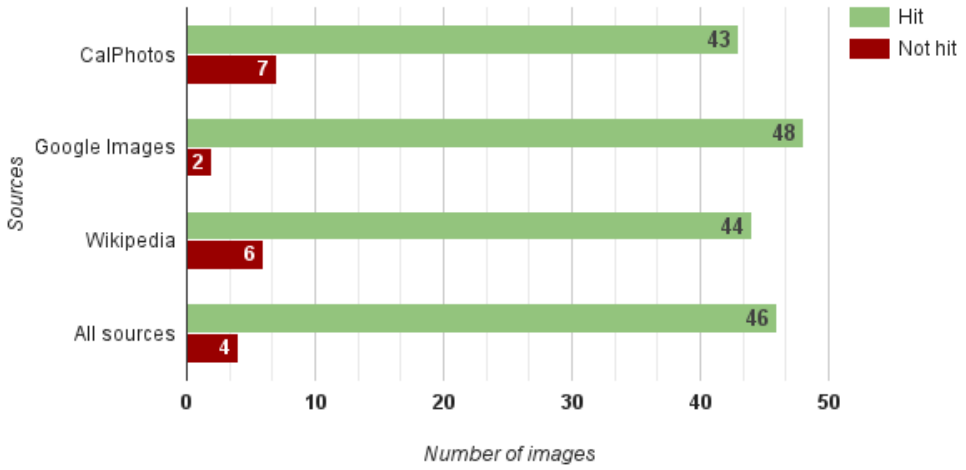

Figure 5. Precision of the image passive extractors.

is done by a supervised technique based on annotated pages used as training. A more recent approach proposes a wrapper induction technique that automatically identifies news articles from web sites. The idea is to used a list of news sites as input and to identify patterns among them. The wrapper than extracts news by seeking these patterns in other web sites [Xiang et al. 2015].

Finally, user-assisted extraction relies on information provided by the user to guide the extraction. In [Meng et al. 2003], the user access a tool to create mappings between an HTML page and a predefined schema. The mappings are transformed into extraction rules represented in XQuery. During extraction, the xquery is executed, transforming the data source into an XML that satisfies the schema. The identification of an object is done by unique elements of the schema. Another related approach is proposed by [Baumgartner et al. 2001]. In this case, the extraction rules are defined in a language called Elog.

Even though all of the above receive external help, they cannot be classified as passive extractors (as we propose) because their main purpose is to find new objects instead of complementing the ones that already exists. Besides, they are tailored for very specific goals: finding relevant data sources (focus crawling), finding objects within a data source (wrapper induction) and uniquely identifying an object as a way to allow deduplication (user assisted extraction). They are unfitted to perform the three things altogether. One the other hand, passive extractors are more naturally suitable for the mentioned tasks, since they focus on single objects at a time.

6. Conclusion

This paper presents an extension for extraction architectures used by materialized data integration systems. The extension classifies extractors and mediators as passive or active. In the active extractor/passive mediator combination, the extractors looks for relevant objects and the mediators is responsible for checking whether the identified objects need to

be merged with already stored objects. Conversely, in the passive extractor/active mediator combination, the extractor receives a message from the mediator asking to complement information about a stored object.

An actual implementation of the architecture was also provided, with the purpose of extracting objects related to animals in order to feed the repository of bioID, a collaborative system for biological classification. The results showed that it is rather straightforward to create passive extractors that find relevant data sources, find objects within data sources and easily identify object duplication.

In comparison with active extraction approaches, the message-driven alternative has the potential to produce extractors that are less complex and more effective. Of course, active extractors are still of great importance. After all, it takes active extractors to feed the repository at the first place. What we state is that passive and active extractors combined constitute a more powerful solution, as passive extractors are able to fill the gaps left by ordinary extraction methods.

References

Anter, S., Zellou, A., and Idri, A. (2016). Retrieving and materializing data in hybrid mediators. *International Journal of Applied Engineering Research*, 11(3):2128–2134.

Baumgartner, R., Flesca, S., and Gottlob, G. (2001). Visual web information extraction with lixto. In *VLDB*, volume 1, pages 119–128.

Cardoso, P., Peripolli, G., Franzin, F., and Mergen, S. (2015). Ambiente colaborativo para identificação de espécies. In *Anais do XIII Simpósio de Informática (SIRC)*, pages 68–73. Centro Universitário Franciscano.

Dong, H. and Hussain, F. K. (2014). Self-adaptive semantic focused crawler for mining services information discovery. *Industrial Informatics, IEEE Transactions on*, 10(2):1616–1626.

Meng, X., Wang, H., Hu, D., and Li, C. (2003). A supervised visual wrapper generator for web-data extraction. In *Computer Software and Applications Conference, 2003. COMPSAC 2003. Proceedings. 27th Annual International*, pages 657–662. IEEE.

Mora, C., Tittensor, D. P., Adl, S., Simpson, A. G., and Worm, B. (2011). How many species are there on earth and in the ocean? *PLoS Biol*, 9(8):e1001127.

Safran, M. S., Althagafi, A., and Che, D. (2012). Improving relevance prediction for focused web crawlers. In *Computer and Information Science (ICIS), 2012 IEEE/ACIS 11th International Conference on*, pages 161–166. IEEE.

Senellart, P., Mittal, A., Muschick, D., Gilleron, R., and Tommasi, M. (2008). Automatic wrapper induction from hidden-web sources with domain knowledge. In *Proceedings of the 10th ACM workshop on Web information and data management*, pages 9–16. ACM.

Xiang, Z.-L., Yu, X.-R., and Kang, D.-K. (2015). Wrapper induction of news information for feeding to social networking service on smartphone. In *Advanced Communication Technology (ICACT), 2015 17th International Conference on*, pages 292–295. IEEE.