A Focused Web Spider Specialized for Turkish Language

Sercan DAĞDAŞ[1], Göksel BİRİCİK[2], Banu DİRİ[2]
[1]Garanti Technology, [2]Yildiz Technical University Computer Engineering Department
[1]cesercandagdas@gmail.com, [2]{goksel,banu}@ce.yildiz.edu.tr

Abstract

We present a focused web spider specialized for Turkish language. We can select which classes to focus by training the system with web pages of the desired topics. Our flexible architecture gives the opportunities to select which HTML tags to scan, tune their weights, select keywords originating from types, and adjust the minimum length for the keywords. We achieved 70% overall classification accuracy with an evaluation dataset of 12 classes crawled from DMOZ directory. Our system distinguishes the classes with numerous training samples better than the ones with fewer samples. We get better classification accuracies as the training dataset grows bigger.

1. Introduction

Numerous amounts of web sites are being rapidly created nowadays. Not just static content but lots of documents, technical or nontechnical articles, news, pictures, music are being shared on the web, causing the information size to increase dramatically. When information size increases, it gets harder for us to find or reach the desired data. The solution came in form of search engines, which have been built to solve the need to find valuable information.

Former examples of search engines were AltaVista, WebCrawler.com, Lycos, Yahoo, etc. These engines tracked different search strategies. Some were good for a certain type of query; however another type of query was not appropriate for that search engine. Later on 1998, another search engine called “Google” was developed by Sergey Brin and Larry Page. Google enhanced old search engines’ strategies by adding algorithms developed by Brin and Page. These algorithms are generally concerned with data mining, text classification and indexing, they developed their own PageRank algorithm [14].

Search engines work by storing information about web pages. Most important component of a search engine which retrieves content from web pages is called “Spider”. Spider scans and downloads web pages, then compresses and stores them to disk. After the retrieval stage, indexer component indexes these pages in a database and prepares pages to user queries. Data retrieved from web pages are stored in an index database for use in later queries.

It is almost impossible to download and fetch every single page on the web. The strategy to solve this problem in our focused web spider is just like any other hypertext crawler (whether for the web, an intranet or other hypertext document collection). The crawler begins with one or more URLs that constitute a seed set. It picks a URL (Uniform Resource Locator) from this seed set, and then fetches the web page at that URL. The fetched page is then parsed, to extract both the text and the links from the page (each of which points to another URL). The extracted text is fed to a text indexer. The extracted links (URLs) are then added to a URL frontier, which at all times consists of URLs whose corresponding pages have yet to be fetched by the crawler. Initially, the URL frontier contains the seed set; as pages are fetched, the corresponding URLs are deleted [1].

Downloaded web pages can be used for different purposes on demand, including *personal queries,* since web crawlers working on personal computers can harvest documents on behalf of the end users [2]; *building a collection,* as the downloaded web pages may be used for different purposes as well as they are looked up in search engines (e.g., *Bharat* and *Broder* crawlers fetch pages from the Yahoo! search engine continuously to form a dictionary [3]); a*rchiving,* especially taking the huge size of the web into consideration (crawlers for archiving should be used with critical performance optimizations [4]); g*athering statistics for the web,* because deriving interesting and practical statistical results from the enormous dataset of crawled web pages is possible (e.g., the proportion of *404-Page not found* error pages to the total number of crawled pages indicates a statistical result [5]).

The first web crawler robot was coded by Matthew Gray in 1993. Chakrabarti [4, 5], Ehrig [6], Aggarwall [7], Diligenti [8] and Menczer [9] are the pioneers of focused web crawling with their initiative studies.

This paper is organized as follows. In Section 2 we introduce our focused web spider that is specialized for Turkish language. Section 3 addresses experimental results derived from our tests. In Section 4 we discuss the outcomes of our application and conclude.

2. Structure and features of our focused web spider

Focused web spider application has very powerful features thanks to robust and accurate algorithms as well as the easy-to-use user interface. We developed our spider on a notebook machine with 2.0 GHz Intel Core2Duo processor, 320 GB 5400 rpm HDD and 1.5 GB of RAM. In this configuration average downloading and scanning operations for a single web page takes only 3 seconds.

We designed our focused web spider to work with only Turkish language, so classification using other languages is impossible, but architecture is designed so flexible that another language add-on can be implemented in a short time.

Naive Bayes Updateable feature of our application provides more accuracy as the application crawls the web. This means long term usage of our spider will converge to a realistic success rate. Another feature called flexible tag point usage enables to add new tags and scores, which gives the ability to tune the spider to the tags in forthcoming HTML versions.

2.1. Naïve Bayes updatable

Our self-learning algorithm is designed to not only use pre-trained vocabulary and class sets, but also contains an option to use web pages that is categorized in one of our trained classes. This option enables our crawler to learn from its experiences. Every newly categorized web page increases the knowledge base of the spider and further crawling may result in more proper results.

2.2. Naïve Bayes algorithm

We use Naive Bayes -a statistical classification method- for our focused web spider. Naive Bayes uses the combined probabilities of the words and classes to determine the class of a document. Let X be our document that we want to find which class it belongs to. We can label the attributes of X as $X=\{x_1, x2... xn\}$. Assume we have m classes in our dataset as $C= \{C_1, C2, Cm\}$. First we calculate probabilities of X over classes with (Eq 1).

$$P(C_i, X) = \frac{P(X / C_i)P(C_i)}{P(X)} \tag{1}$$

$$P(X / C_i) = \prod_{k=1}^{n} P(x_k / C_i) \tag{2}$$

Instead of calculating $P(X/C_i)$, we can use (Eq 2) in order to reduce the processing cost on calculations, assuming the attribute values of X are independent.

It will be enough to calculate only the nominator part of (Eq 1), since the denominator is always the same for X. The class of X is predicted by choosing maximum a posteriori classification (MAP) value out of the calculated probabilities using (Eq 3) [12].

$$C_{MAP} = \arg\max_c \prod_{k=1}^{n} P(x_k / C_i) \tag{3}$$

2.3. URL search

We search URL's of web sites as a focusing hint in our focused web spider. This is especially useful for focused crawling, because keywords we desire generally lies in URL's of web sites. Since the web pages on sites are more frequently generated by automated tools depending on the content, URL's are shaped with the topic keywords of the page, delimited with underscore (_) or dash (-) characters. This keyword information is essential to categorize a web page accurately, even if only scanning URLs do not give attention to its content. Search engines usually utilize general purpose crawlers that scan URLs for a hint that may be a candidate to reside along the top side of retrieved results. This search strategy affects web sites' position in overall search results. Searching URLS of web sites also gives our focused web spider more chance to detect the keywords exposed especially by this kind of sites.

2.4. HTML tag usage

Another challenging feature of our focused web spider is using HTML tags as keys to achieve more accurate classification results. We know that web masters usually place their keywords in more important tags like <*title*>, <*meta*>, <*h1*>, <*h2*>, <*h3*>, etc. Our spider gives us option to define tag weights, which are multipliers for HTML tags in training phase. Users are also allowed to define their own tags to fine tune the training parameters in order to achieve even higher classification accuracy. Our spider uses a database to store the training information about the classes and keywords. Classes table is a basic header table to keep information for the trained classes.

2.5. Keywords table and HTML tag weight calculation

Keywords table holds the root, count and multiplier data of keywords. We only insert new records if the root of the keyword is valid and its count meets the defined criteria. The count field holds information about how many times this root keyword is found in this classes' related documents. For example, the root for the words *kedim* and *kediler* is *kedi*. This keyword is hold in a record in keyword table like "*classId: 1, rootKeyword: kedi, wCount: 2, multiplier: 1*".

The multiplier field holds information about a word's importance in documents of the training set. Let us continue our example. Assume we have a HTML document in our training set, with a title tag like "*<title> Kedi resimleri </title>*". If we specify the weight of *<title>* tag as *16*, multiplier field is calculated using simple averaging as given in (Eq 4). μ resembles our *multiplier* and χ symbolizes *wCount*.

This field is the crucial key for our focused web spider to tune importance of HTML tags to categorize documents into classes. Careful and intensive balancing of tag weights results in more accurate classification.

$$\mu_{new} = \frac{\chi \times \mu_{old} + \mu_{new}}{\chi + 1} \Rightarrow \frac{(2 \times 1) + 16}{3} = 6 \quad (4)$$

2.6. Respect to robots exclusion protocol

As noted by Koster, since the use of web crawlers is useful for a number of tasks, there are also costs that one must face. The charges of using Web crawlers include [12] network resources, as crawlers require considerable bandwidth and operate with a high degree of parallelism during a long period of time; server overload, especially if the frequency of accesses to a given server is too high; poorly-written crawlers, which can crash servers or routers, or which download pages they cannot handle; personal crawlers that, if deployed by too many users, can disrupt networks and Web servers.

A partial solution to these problems is *the robots exclusion protocol*, also known as the *robots.txt* protocol that is a standard for administrators to indicate which parts of their web servers should not be accessed by crawlers. This standard does not include a suggestion for the interval of visits to the same server, even though this interval is the most effective way of avoiding server overload.

Our focused web spider has respect to web sites' crawling policies. We read the *robots.txt* file found in the root folder of a web site's URL and do not retrieve pages indicated in robots exclusion list of that web site. This policy also may prevent the focused spider to enter an infinite loop generally caused by the query string variables, known as spider traps. Since our crawler do not have a world-known name, our focused web spider do not control *useragent* section for a certain name such as *google-bot* but looks for a line like "*user-agent=**" that indicates all spiders including this one should obey specified rules. A fragment of the robots.txt file of *Google* web site is as follows:

```
User-agent: *
Allow: /searchhistory/
Disallow: /search
Disallow: /catalogs
```

2.7. Multi threading

Our focused web spider application has two separate threads to increase the crawling speed and make efficient use of hardware resources. A single threaded crawler may lose critical times when waiting for a page to be downloaded. During the downloading time period, the central processing unit remains idle. We know that network I/O can be more time consuming than disk I/O. That's why our spider uses another thread to make calculations for classification. In this architecture, our second thread continues working on scheduled tasks when the first thread waits for I/O.

3. Experimental Results

In this section we present our experimental results. First we introduce our evaluation dataset. Then we explain our testing parameters. Finally we discuss the test results.

3.1. Evaluation Dataset

Evaluation dataset is prepared from web pages under DMOZ [15] category World/Turkce. World/Turkce category included 12342 web pages in Turkish at the time we crawled. There are twelve main categories in this set: *Shopping*, *News*, *Computers*, *Science*, *Recreation*, *Business*, *Reference*, *Arts*, *Games*, *Health*, *Sports* and *Society*. For our test on heterogeneous dataset, we used whole pages of our crawl. We split 33% of pages in each class for testing, and 67% for training. The train-test splitting of the dataset, class labels and number of keywords obtained by the training of the system with the Naïve Bayes classifier mentioned in Section 2.2 are given in Table 1.

To test our system on a homogeneous dataset, we randomly choose 200 pages from each class and split 150 for training, and 50 for testing. Number of keywords obtained by the training of the system with the Naïve Bayes classifier mentioned in Section 2.2 is given in Table 2. Note that this time we have 150 pages for training and 50 pages for testing in each class.

Table 1. Samples in the heterogeneous dataset

Class Labels	# pages train set	# pages test set	# key-words
Shopping (Sh)	368	184	213
News (N)	328	149	1432
Computers (C)	1413	706	2406
Science (Sc)	231	115	1339
Recreation (Rc)	378	189	1677
Reference (Rf)	518	259	1267
Arts (A)	616	307	2195
Games (G)	133	66	569
Health (H)	585	292	1920
Sports (S)	274	137	1247
Society (Soc)	777	388	2817
Business (B)	2622	1307	4320

Table 2. Number of keywords per class for the homogeneous dataset

Class Labels	# key-words	Class Labels	#key-words
Shopping (Sh)	1365	Arts (A)	879
News (N)	949	Games (G)	612
Computers (C)	696	Health (H)	936
Science (Sc)	904	Sports (S)	870
Recreation (Rc)	983	Society (Soc)	898
Reference (Rf)	565	Business (B)	977

3.2. Testing parameters for evaluation

As mentioned in Section 2.4 and 2.5, we can select which tags to use form the input pages in training phase, as well as we can tune their weights. Tags we chose to train the system for our experimental tests and their tuned weights are given in Table 3.

Our spider lets us to set the minimum keyword length. We set this value to *3*, in other words we omit *1* and *2* letter words. Another parameter that we can arrange is the type of keywords. We may choose verbs, nouns, abbreviations, adjectives, pronouns, exclamations or pronouns. In our tests we only choose noun type keywords and excluded other types.

Table 3. Chosen tags and their tuned weights for experimental tests

Tag name	Weight (Score)
URL	100
<title>	20
<meta>	10
<h1>	8
<h2>	4
<h3>	3

3.3. Test results

We use three different metrics to quantify our test results. The calculation of precision, which is also known as *harvest rate* and recall, is given in (Eq 5). Macro-average F Measure is calculated with (Eq 6). *C* is the number of classes. *tp* is the number of true positives, *fp* is the number of false positives and *fn* is the number of false negatives for the samples predicted. The confusion matrix of true/false positive/negative predicted samples is given in Table 4.

$$p = tp/tp + fp\,, r = tp/tp + fn \qquad (5)$$

$$F_i = 2 \times \frac{p_i \times r_i}{p_i + r_i}, F_{macro} = \sum_{i=1}^{C} F_i \Big/ C \qquad (6)$$

Confusion matrices of the test results are given in Table 5 and Table 6 sequentially. Looking at the recalls, we observe that *computers* and *business* are classified most successfully, whilst *shopping* and *news* are the worst classified ones. Results show that *shopping* interferes with *business*, and *news* interferes with *society*. The heterogeneous dataset gives better results (0.71 F-measure, 70% overall accuracy) than the homogeneous dataset (0.59 F-measure, 59% overall accuracy) because it has much more training samples.

Table 4. Confusion matrix for predictions

		Actual class	
		C1	C2
Predicted class	C1	tp	fp
	C2	fn	tn

4. Conclusions

In this paper we present a focused web spider, which is specialized for Turkish language. It is possible to select which classes to focus by training the system with a dataset of web pages in the desired topics. Our flexible architecture gives the opportunities to select which HTML tags to scan, and tune their weights for the classes. Moreover the users can select keywords originating from verb, noun, abbreviation, adjective, pronoun, or exclamation types, and adjust the minimum length for the keywords.

On our evaluation tests we utilize a dataset crawled from DMOZ directory in 12 classes. We achieved 70% overall classification accuracy and 0.71 macro averaged F measure with this dataset. We see that our system distinguishes the classes with numerous training samples better than the ones with fewer samples. One gets better classification accuracies if the training set is bigger.

5. References

[1] C. D. Manning, P. Raghavan and H.Schütze, "An Introduction to Information Retrieval", *Cambridge University Press,* 2009.

[2] M. Chau, D. Zeng and H. Chen, "Personalized Spiders for Web Search and Analysis", *Proc.the 1st ACM-IEEE Joint Conference on Digital Libraries*, Roanoke, Virginia, USA, pp. 79-87, 2001.

[3] M.R. Henzinger, A. Heydon, et-al., "On Near-uniform URL Sampling", *Proc. the 9th Intl. World-Wide Web Conference* ,Amsterdam, Netherlands, 2000.

[4] B. Kahle, "Preserving the Internet", *Scientific America*, 1997.

[5] M. Gray, "Internet Growth and Statistics: Credits and Background", http://*www.mit.edu/~mkgray/net/ background.html*, 1993.

[6] S. Chakrabarti, M. Berg, and B. Dom, "Focused Crawling: A new Approach to Topic-specific Web Resource Discovery", *In Proceedings of the 8th International WWW Conference*, Canada, 1999.

[7] S. Chakrabarti, K. Punera and M. Subramanyam, "Accelerated focused crawling through online relevance Feedback", *In WWW*, 2002.

[8] M. Ehrig and A. Maedche, "Ontology-focused Crawling of Web Documents", *In Proceeding of the 2003 ACM Symposium on Applied Computing*, 2003.

[9] C. Aggarwal, F. Al-Garawi and P. Yu, "Intelligent Crawling on the World Wide Web with Arbitrary Predicates", *In Proceedings of the 10th International World Wide Web Conference*, Hog Kong, 2001.

[10] M. Diligenti, F. Coetzee, S. Lawrence, C. Giles and M. Gori, "Focused Crawling Using Context Graphs", *In Proceedings of the 26th International Conference on Very Large Databases*, Cairo, 2000.

[11] F. Menczer, G. Pant, P. Srinivasan and M. Ruiz, "Evaluating Topic-driven Web Crawlers", *In Proceedings of the 24th Annual International ACM/SIGIR Conference*, USA, 2001.

[12] T. M. Mitchell, Machine Learning, *Mc.Graw-Hill*, 1997.

[13] M. Koster, "A Standart for Robot Exclusion", *http://www.robotstxt.org/wc/norobots.html*, 1994.

[14] L. Page, S. Brin, M. Rajeev and T.Winograd, "The pagerank citation ranking: Bringing order to the web.", *Technical report, Stanford InfoLab*, 1999.

[15] Directory Mozilla Project, *http://www.dmoz.org,* 2009.

Table 5. Confusion matrix of the results using the heterogeneous dataset

		Actual												
		Sh	**N**	**C**	**Sc**	**Rc**	**Rf**	**A**	**G**	**H**	**S**	**Soc**	**B**	***Recall***
Predicted	**Sh**	**82**	3	18	1	4	4	10	3	9	7	5	38	0.45
	N	4	**65**	13	4	2	1	14	1	1	3	30	10	0.44
	C	10	8	**608**	0	3	10	14	8	0	3	14	27	0.86
	Sc	0	0	6	**79**	0	15	3	0	1	1	7	3	0.69
	Rc	6	4	13	1	**100**	3	17	1	3	10	11	20	0.53
	Rf	0	0	15	8	0	**205**	2	0	5	3	15	6	0.79
	A	1	3	9	4	2	15	**248**	1	1	2	16	4	0.81
	G	0	2	7	0	0	0	0	**52**	1	1	0	3	0.79
	H	4	4	10	6	0	16	5	1	**203**	2	21	20	0.70
	S	1	0	6	0	3	2	1	0	2	**112**	3	7	0.82
	Soc	3	7	39	8	0	14	21	2	0	6	**273**	14	0.70
	B	23	5	86	7	3	11	22	3	9	9	21	**1108**	0.85
Precision		0.61	0.64	0.73	0.67	0.85	0.69	0.69	0.72	0.86	0.70	0.66	0.88	

Table 6. Confusion matrix of the results using the homogeneous dataset

		Actual												
		Sh	**N**	**C**	**Sc**	**Rc**	**Rf**	**A**	**G**	**H**	**S**	**Soc**	**B**	***Recall***
Predicted	**Sh**	**29**	3	4	1	6	1	2	0	3	0	0	1	0.58
	N	5	**29**	0	1	1	1	0	0	1	4	6	2	0.58
	C	2	3	**29**	1	0	3	0	3	0	1	4	4	0.58
	Sc	0	2	1	**29**	1	7	2	1	2	1	4	0	0.58
	Rc	3	1	0	1	**35**	1	4	0	0	3	1	1	0.70
	Rf	1	0	2	11	0	**33**	1	0	0	1	1	0	0.66
	A	4	3	1	4	2	2	**31**	0	0	0	1	2	0.62
	G	2	3	2	1	0	0	2	**37**	0	2	1	0	0.74
	H	2	2	0	3	0	2	0	1	**35**	1	3	1	0.70
	S	5	2	1	1	6	0	1	4	1	**27**	2	0	0.54
	Soc	0	3	4	7	1	3	5	2	4	1	**19**	1	0.38
	B	12	0	6	1	1	1	4	0	0	1	1	**23**	0.46
Precision		0.45	0.57	0.58	0.48	0.66	0.61	0.60	0.77	0.76	0.64	0.44	0.66	