A Cleaning Algorithm for Noiseless Opinion Mining Corpus Construction

Otman Manad, Anna Pappa, Gilles Bernard

To cite this paper : O. Manad, A. Pappa and G. Bernard, "A Cleaning Algorithm for Noiseless Opinion Mining Corpus Construction," 2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA), 2018, pp. 1-7, doi: 10.1109/AICCSA.2018.8612867.

A Cleaning Algorithm for Noiseless Opinion Mining Corpus Construction

Otman Manad Anna Pappa Gilles Bernard

***Abstract*—This paper presents DyCorC (Dynamic Corpus Constructor), an extractor and cleaner of web forums contents. Its main points are that the process is entirely automatic, language-independent, without any previous knowledge, and adaptable to all kinds of forum architectures. The corpus is built accordingly to user requests and minimizes the boilerplate for further feature-based opinion mining and sentiment analysis. Such noiseless corpora are usually hand made with the help of crawlers and scrapers, devised for each type of forum specific containers, which entails lots of work and skills. Our aim is to cut down this preprocessing stage. Our algorithm is compared to state of the art models (Apache Nutch, BootCat, JusText), with a gold standard corpus we released. DyCorC offers a better quality of noiseless content extraction so that automatically could create new thematic corpora. The algorithm is based on DOM trees with string distances, seven of which have been compared on the reference corpus, and feature-distance has been chosen as the best fit.**

***Index Terms*—Web data cleaning, Noiseless corpora construction, Boilerplate detection**

I. Introduction

Building a corpus from web forums and blogs is a mandatory first step for opinion mining and sentiment analysis. Since the expansion of the digital interaction between users and social networks, the need for such corpora has exploded. But those corpora are cluttered with all sorts of noise (boilerplate, ads, duplication and so on), which make them unusable without preprocessing, and it is necessary to re-organize data cleaning, editing and matching procedures. Besides, relatively frequent website changes require re-programming of the web content extractor by skilled IT programmers [1], [2]. Our aim is to cut down and ultimately suppress this preprocessing stage.

We present here an unsupervised content extractor, DyCorC (Dynamic Corpus Constructor). The algorithm embedded in DyCorC parses the DOM page structure and uses a string distance to detect the leaves that contain the relevant content (posts, reviews or comments). Seven string distances (Levenshtein, Damerau-Levenshtein, Feature-Distance, Jaccard, Jaro, Winkler-Jaro, Longest Common Subsequence) were compared for this task, and the best ones are chosen. Our system crawls the web forums with a list of seed urls, detects the duplicate pages, and retrieves the relevant content.

This algorithm has been compared with three state of the art models: Apache Nutch, BootCat and JusText, a task for which we devised a gold standard corpus.

This paper is organized as follows. Section 2 describes the issue, section 3 gives an overview of related works. Section 4 describes the system in detail. Section 5 presents the experiments and section 6 concludes our work and proposes further suggestions for future work.

Fig. 1. Relevant and noisy content in a web forum

II. The issue

The main issue we consider here is building a corpus from web forums and blogs, correctly determining the relevant content in crawled web pages, without user supervision nor knowledge about the page language or structure. Irrelevant content including ads, information about the website (general information, website map, akin websites...), links to most recent posts or similar posts, sign-up frame, share with (Twitter, Facebook...) frame, navigation menus, and so on, is cleaned. In figure 1 the blue part is the relevant content, the rest is noise. Measured in number of words, the proportion of relevant content usually approximates to 50% and sometimes down to 10%.

If the corpus is not thoroughly cleansed, it is unusable, because later processing is slowed and flawed by the frequency of boilerplate items, which can be very repetitive and even not in the same language than the relevant content.

Up to now, the most efficient methods for cleaning are manual, supervised or semi-supervised methods [3] that let the user define containers or wrappers. This is done either by way of selection of relevant content within an API or by the devising of a regular expression or a program (e.g. the devising of a Python spider in Scrapy [4]). The main drawback of these methods are that the results are specific to the parsed forum and, at most, to forums of the same structure. There are numerous Content Management Systems for forum building, each one producing its own page structure. Thus the process of building a corpus from different web forums is a

time-consuming and hard task, that needs good skills in html structure or programming [5].

Hence the need for a completely unsupervised system that could build such a corpus without any data given by the user except the research query (urls or keywords). It should be language-independent as well as independent from the forum architecture.

A secondary but important issue is that there are very few easily available multilingual gold standard corpus in this field. We found only three of them: TBDW[1] (Testbed for Information Extraction from Deep Web) done by Hirokawa lab from Kyushu university, in English and Japanese, CleanEval 2007, done by the challengers, in English and Chinese, and the Bolt corpus [6], a very big forum corpus (1,597,500 XML pages), in English, Chinese and Egyptian Arabic, of which only a very small subset has been manually processed to this day.

TBDW contains only one data record from each web page, which strongly affects its usability. CleanEval does not contain any forum, and there are doubts expressed by Kohlschütter (the author of Boilerpipe) as to whether it is "appropriate for the purpose of boilerplate detection" [7]. The processing of Bolt, especially the manual filtering, is not advanced enough. Thus we had to devise and release our own gold standard corpus for forum content extraction, available at

III. Related work

In the last years quite a number of crawling and extracting systems have been devised. An overview of such techniques is detailed in [8], which advocates Focus [9], a supervised web forum crawler, which uses partial tree alignment and according to its authors outperforms the others. As another example, [10] creates large-scale corpora from Twitter for events detection evaluation, while [11] propose a scalable model with ontology techniques for relevant content extraction.

Methods for extracting clean contents from forums can be classified according to:

- data they work on: the text, the html structure, or both;
- number of pages as input: page by page or set of pages;
- knowledge used: about the html tags (e.g. tags most probably used for boilerplate, tag density, url structure), about text (word density, stopwords, shallow syntax);
- whether they look at the architecture of the page or DOM tree for detecting the most probable location of the relevant content.

One of the first unsupervised forum web noiseless crawler is Roadrunner (references and brief description).

It works on the html tree (an equivalent of DOM tree), taking a set of pages as input and and comparing their trees with simple tree matching algorithm. The relevant content is determined by ********** ?

The well-known state of the art unsupervised systems, easily available for researchers as they are open-source and maintained by large teams, are Apache Nutch and BootCat.

[1]http://daisen.cc.kyushu-u.ac.jp/TBDW/, consulted 04/04/2017

Apache Nutch (reference), implemented in Java, is divided into four components: crawler, indexer, database storage, fetcher. SolR server is used to read the indexed data. For boilerplate, it uses the library Boilerpipe [7] that has been devised to extract the content of news sites. This library works on text and html tags, page by page, using knowledge on both: uppercase letters, length of words, beginning and length of sentences (which implies knowing about sentence structure), keywords (for boilerplate), a list of probable boilerplate tags, number of links in A tags, etc. The version included in Nutch only uses word and tag densities in branches of the html tree. The relevant content is where word density is higher and tag density lower.

BootCat (reference) is based on search engine results and regular expression techniques; a major drawback is that its crawler does not control if an url has already been crawled. For boilerplate, it works on both text and html tree, page by page, using knowledge on the language (a stopword list), and a ngram language model. It counts the number of tags in nodes and compares it to the number of ngrams. Following somewhat the same principle than Nutch, the relevant content is where ngrams are higher and tags lower.

More recent models with unsupervised techniques for forum content extraction include the RevScrap system [12] which works on the DOM tree, page by page, with knowledge about the date format in regular expressions. It looks for the branch that contains the date. This very simple heuristic gives interesting results, but no information about noise is available, and this method does not work for Arabic or Chinese date format.

********** some other examples here

More to our point and easily available[2], JusText [] works on both text and html structure, page by page, using knowledge on the language (a stopword list) and on the html tags. It counts the number of words and the number of stopwords in leaves, and uses a list of probable boilerplate tags.

IV. System description

Our system crawls web forums and extracts the source code. It is language-independant, no linguistic knowledge being used; no knowledge about tags either. The parsing is done page by page. We work on both text and html structure, and use the DOM tree, comparing its branches in order to locate the most probable location of relevant content. The html structure, after being cleaned from html errors and inconsistencies by Tidy Html[3] of W3C, is converted in a DOM tree [13], saved in XML format. The extraction of relevant content can be done immediately, on the DOM tree, or later, by reading the XML file; it is in his turn saved in an XML file.

DyCorC is written in C++ using Qt5 libraries, build with Cmake, running on *nix systems, and it is available at "gitlab.com/Data-Liasd/ForumCorpusSelect".

********** Detailed schema of the crawling and parsing (not only crawling) of DyCorC, 2 is not good.

[2]http://corpus.tools/wiki/Justext

[3]http://tidy.sourceforge.net/

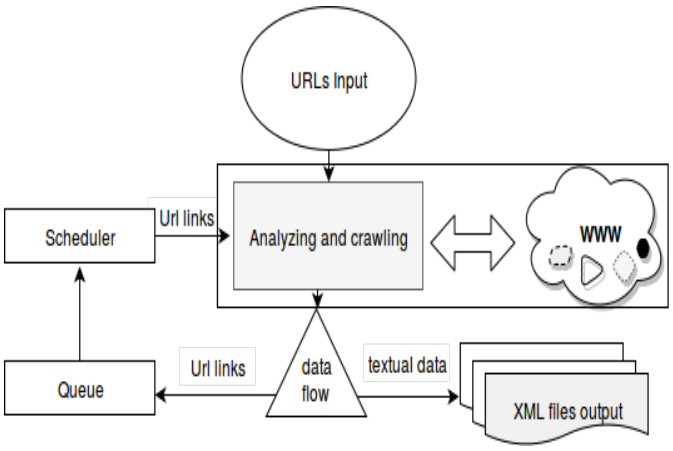

Fig. 2. Overview of the crawler

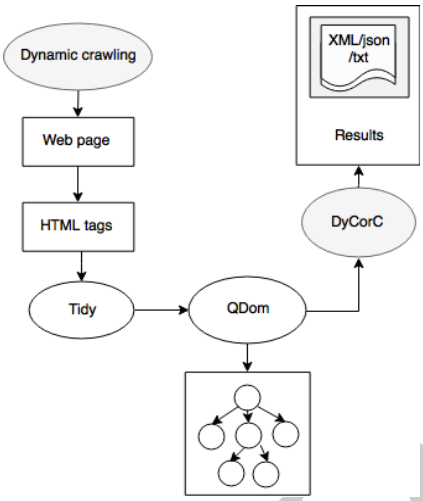

Fig. 3. Dynamic crawling

We will not dwell on the crawling part, as it does not have any special feature. Just note that, as Nutch or others, we follow politeness rules, use multithreading and detect duplicated pages with checksum; and that we gather data from url seeds, either web urls or local ones. Our main contribution is on the content extraction algorithm, which is detailed below.

We use the DOM tree as input, to be separated into data regions, ultimately relevant and noisy content. The general principle of our method is that the most recurring structures containing the most diverse texts are more likely to contain relevant content. In a first stage, we detect the similar nodes in the DOM tree and put them in categories, not taking the text nor the attributes into account. Every node is represented by its path from the root, as in $\{< body >< h1 >< p >< span >\}$, converted into a string, as in "bodyh1pspan". In fact we omit the *body* tag, as it is part of every path. Similarity is detected in the following way.

For every pair of nodes of one level in the DOM tree, we compare their strings s_i and s_j with a string distance. If this distance is lower than the average length of their strings

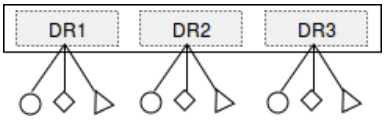

Fig. 4. Data records of the same category

(inequation 1), they are in the same category. We recurse on every level. The resulting categories contain similar blocks as in figure 4.

$$Dist(s_i, s_j) < \frac{length(s_i) + length(s_j)}{2} \tag{1}$$

Then we will compute the mean textual distance by category, which will give us an measure of text diversity in recurring blocks: in each category c_k, we compute pairwise string distance between textual content of the branches, sum all distances and divide by the number of pairwise combinations (equation 2).

$$MeanDist_T(c_k) = \frac{\sum_{i \neq j}^{\forall b_i, b_j \in c_k} Dist_T(b_i, b_j)}{C_2^{|c_k|}} \tag{2}$$

The first stage has usually left us with some singletons, which we try and incorporate into the existing categories with the following process.

For each category c_k we select the node n_i whose average textual distance to the others is the closest to the mean textual distance of c_k (equation 3). This node will be the representant of c_k.

$$repr(c_k) = \min_{i=0}^{|c_k|-1} \left(MeanDist_T(c_k) - \frac{\sum_{j \neq i}^{j<|c_k|}}{|c_k| - 1} \right) \tag{3}$$

Then we take into account the attributes Att_S of the singleton S and of all representants, and compute for each one the Jaccard index of the attributes. The singleton will go into the category with the highest Jaccard index, if it is greater than a user fixed threshold θ (inequation 4, N being the total number of categories).

$$\left(\max_{k=0}^{N-1} \frac{|Att_S \cap Att_{repr(c_k)}|}{|Att_S \cup Att_{repr(c_k)}|} \right) > \theta \tag{4}$$

After integration of a singleton, we recompute the mean textual distance of the category. This process is recursively applied until no more integration can take place.

Last we determine which category contains the richest textual content, that is, the one whose mean text distance is higher (equation 5). The leaves of this category are the post contents.

$$\max_{k=0}^{N-1} MeanDist_T(c_k) \tag{5}$$

In a postprocessing stage, we retrieve dates and reviewers identities from the post content with regular expressions.

The string distances and similarities we have implemented are: Levenshtein, or edit distance, based on the number of character insertion, deletion and substitution operations necessary to convert one string to the other; Damerau-Levenshtein, adding transpositions to Levenshtein; Jaro [**?**], based on the number of identical characters whose index are closer than half the longest string size, and on the number of transpositions; Jaro-Winkler, which gives more weight to the beginning of strings than to the end; Feature-distance, based on the number of common ngrams; Longest Common Subsequence or LCS, whose name is transparent; Jaccard distance, based on the ratio between the cardinals of the intersection and the union of two strings.

@INPROCEEDINGSKondrak05n-gramsimilarity, author = Grzegorz Kondrak, title = N-gram similarity and distance, booktitle = Proc. Twelfth Intl Conf. on String Processing and Information Retrieval, year = 2005, pages = 115–126

V. Experiments

We have conduced two series of evaluation on two criteria. The criteria were: (a) computing time (b) quality of extraction (or noise cleaning). The quality of extraction is measured with precision, recall and F-measure. Taking W_f as the number of found words and W_g as the number of ground truth words, *precision* is $\frac{W_f \cap W_g}{W_f}$, *recall* is $\frac{W_f \cap W_g}{W_g}$ and *F-measure* the harmonic mean of both.

The first series of evaluation was between the various string distances implemented, in order to choose the best one according to both criteria. The second series of evaluation is between our system, DyCorC, and three state of the art systems, Apache Nutch, BootCat and JusText.

For these tests we devised a specific gold standard corpus, for the reasons given in the issue section, presented in next subsection. The following subsection will present the experiments on distances, with computing time and with quality of extraction. Next one will compare the aforesaid four systems.

A. Gold Standard Corpus

The gold standard used in this paper is described in table I. The chosen websites do not have any *robots.txt* file. We have chosen to have roughly the same number of words in all four languages (between 716,000 and 779,000).

The experiments given here are computed with 8 threads on a PC. The average internet reception rate of the experiments was around 3 Mbps.

B. Comparing string distances

The table II gives the average values of our criteria for each distance. The time, given in seconds by page, is the mean extraction time (without crawling) done with 8 threads, computed only for the pages where relevant content was found.

Behind these average values, however, very different situations are to be found. For instance, on some forums, some distances perform very badly and very quickly. In a general manner, the architecture of the pages plays an important role in the computation and its results. For lack of place, we will only explore three typical cases.

TABLE I
Gold standard corpus

	domain	pages	words	relevant words
French	developpez.com	33	226 327	33 489
	ubuntu-fr.org	47	300 364	201 126
	etudes-litteraires.com	113	226 595	79 184
	Total French	**193**	**753 286**	**313 799**
Greek	ubuntu-gr.org	85	366 284	122 843
	dotnetzone.gr	57	195 153	41 805
	fe-mail.gr	5	203 136	139 252
	Total Greek	**147**	**764 573**	**303 900**
English	englishforums.com	366	716 276	186 886
Arabe	forum.ency-education.com	238	715 572	71 192
Total		**944**	**2 949 707**	**875 777**

TABLE II
Comparing average values for distances

distance	recall	precision	F-measure	time
Feature distance	91.76%	93.75%	**91.25%**	0.026
Levenshtein	91.76%	93.70%	91.22%	0.742
Damerau-Levenshtein	91.76%	93.70%	91.21%	1.1
Jaro	56.38%	57.29%	56.54%	0.043
Jaro-Winkler	56.38%	57.29%	56.54%	0.037
LCS	56.38%	56.90%	56.63%	0.03
Jaccard	49.46%	56.22%	48.51%	0.06

The first case (the most frequent) is illustrated in table III: the best distance is Feature-distance followed closely by Levenshtein and Damerau-Levenshtein. LCS is quicker but did not find any relevant content.

TABLE III
Average time: Feature-distance followed by Levenshtein

Domain	Distance	F-measure	Time (s/p)
developpez.com	Levenshtein	95.94%	0.82
	Damerau-Levenshtein	95.94%	1.13
	Feature-distance	**96.17%**	**0.18**
	Jaro	-	1.5
	Jaro Winkler	-	1.25
	Jaccard	-	1.42
	LCS	-	0.003

The table IV presents another case, where Feature-distance is quicker, followed by LCS distance, Levenshtein and Damerau-Levenshtein being the slowest distances. Quality of extraction is the same for all distances. In other instances, Jaro can be the second best.

But the case presented in table V is much more interesting: there Jaro distance performs as well as any other and takes much less time than Feature-distance. In some of these cases, the quality of extraction is even better than Feature-distance.

All in all, Levenshtein and Damerau-Levenshtein perform quite as well as Feature-distance, but take nearly always longer. Jaro-Winkler has the same results than Jaro, and takes

TABLE IV
AVERAGE TIME: FEATURE-DISTANCE NOT FOLLOWED BY LEVENSHTEIN

Domain	Distance	F-measure	Time (s/p)
ubuntu-gr.org	Levenshtein	97.45%	0.014
	Damerau-Levenshtein	97.45%	0.016
	Feature-distance	97.45%	**0.004**
	Jaro	97.45%	0.012
	Jaro Winkler	97.45%	0.012
	Jaccard	97.45%	0.008
	LCS	97.45%	0.005

TABLE V
AVERAGE TIME: JARO DISTANCE FIRST

Domain	Distance	F-measure	Time (s/p)
etudes-litteraires.com	Levenshtein	94.20%	0.011
	Damerau-Levenshtein	94.20%	0.014
	Feature-distance	94.20%	0.01
	Jaro	94.20%	**0.002**
	Jaro Winkler	94.20%	0.002
	Jaccard	94.20%	0.02
	LCS	94.20%	0.01

more or less the same time. Jaccard never is best, LCS is worse or equal to Jaro in quality and once was better in time.

The most interesting distances, according to a good measure of tests, are Feature-distance, as it is the best in quality and it has on average the shortest computing time, and Jaro distance, as in some cases it divides the extraction time by 5 without loss of information, or even with better quality results. Their detailed results are presented in table VI and VII.

TABLE VI
FEATURE-DISTANCE QUALITY

Domain	lang.	recall	precision	F-measure	Time
developpez.com	french	100%	92,61%	96,17%	0.18
ubuntu-fr.org	french	100%	92,28%	95,99%	0.008
etudes-litteraires.com	french	100%	89,04%	94,20%	0.01
ubuntu-gr.org	greek	95,03%	100%	97,45%	0.004
dotnetzone.gr	greek	100%	95,22%	97,55%	0.02
fe-mail.gr	greek	100%	95,40%	**97,65%**	**0.0002**
englishforums.com	english	43,68%	100%	60,80%	0.03
forum.ency-education.com	arabic	95,44%	85,50%	90,20%	0.02

The only forum where Feature-distance has a bad recall is *englishforums.com*, and it is the only one where Jaro distance is best in quality. No knowledge about language being included in our algorithm, the only possible influence is the forum architecture.

As one can see, Jaro distance either performs as well (or better) than the other distances, or does not perform at all. The factor seems to be the length of the paths in the DOM tree. The same holds for Jaro Winkler and LCS distances and partly for Jaccard distance. On the 944 pages of our gold standard corpus, correctly analyzed by Feature-distance, Levenshtein and Damerau Levenshtein, Jaro and Jaro Winkler

TABLE VII
JARO DISTANCE QUALITY

Domain	lang.	recall	precision	F-measure	Time
developpez.com	french	0%	0%	-	1.5
ubuntu-fr.org	french	0%	0%	-	0.002
etudes-litteraires.com	french	100%	89,04%	94,20%	0.002
ubuntu-gr.org	greek	95,03%	100%	97,45%	0.012
dotnetzone.gr	greek	100%	95,22%	97,55%	0.0025
fe-mail.gr	greek	100%	95,40%	**97,65%**	**0.0004**
englishforums.com	english	56,07%	78,72%	65,49%	0.07
forum.ency-education.com	arabic	0%	0%	-	0.07

only processed 626 pages, LCS 668 pages, and Jaccard 260.

Our conclusion provisorily was to embed Feature-distance in our algorithm, as it is stable in quality and it has on average the shortest computing time. But it would be interesting to make a guess on the forum architecture, looking at the DOM tree of one or two pages before choosing between Feature-distance and Jaro distance.

C. Compared evaluation

We have selected Apache Nutch [14] (version 1.12) with SolR server in 4.10 version), BootCat [15] (stable version 2014) and JusText as state of the art models. Let us begin by comparing their average values (not weighted) to DyCorC ones (with Feature-distance) in table VIII. As on one side JusText does not crawl the data and on the other side BootCat and Nutch do not separate crawling time from extraction time, we patched JusText to our crawler (on 8 threads) in order to have a comparandum.

TABLE VIII
AVERAGE VALUES OF THE MODELS

Model	precision	recall	F-measure	time (s/p)
Nutch	61.53%	75.03%	52.5%	1.61
BootCat	36.73%	96,91%	49.19%	0,85
JusText	43.76%	**100%**	58.29%	1.26
DyCorC	**93.75%**	91.76%	**91.25%**	**0.77**

Table IX gives the global values for the processing of the whole of the gold standard corpus for each model (words retrieved are ground truth words).

TABLE IX
GLOBAL RESULTS

	Nutch	BootCat	JusText	DyCorC
Time (secs.)	1581,9	**390,44**	911.07	468.15
Words retrieved	480 257	266 760	361 163	**800 932**

JusText has the best recall (it found all ground truth words on the whole gold standard corpus), followed by BootCat, DyCorC with 91.76% of the ground truth words being found, Nutch having the worst recall. DyCorC has the best precision, with 93.75% of found words being ground truth words,

followed by Nutch with 61.53%; both JustText and BootCat having bad precision (less than one found word in two being ground truth). Thus globally DyCorC is the best in quality of extraction; it retrieves more than 800,000 ground truth words in the gold standard (875,777 total).

As for time, BootCat is the best on the whole gold standard, followed by DyCorC, JusText and Nutch having the worse time. DyCorC has the best average time, but as the average values are not weighted by the number of pages of each domain, this only means that DyCorC is the less influenced by the forum structure, while BootCat (table XI) has been impeded by *fe-mail.gr*, a very small domain.

Looking more closely at the results given in tables X, XI, XII, one can see also that it is the arabic forum (*forum.ency-education.com*) that gave Nutch its bad result. Without it, Nutch is the quicker algorithm (followed by BootCat). Whether it is due to the structure or to the language or both is hard to say.

The speed of BootCat is remarkable as it is not multithread, while Nutch and DyCorC were both on 8 threads.

TABLE X
NUTCH

Domain	lang.	precision	recall	F-meas.	time
developpez.com	fr	49.53%	100%	66.25%	1.23
ubuntu-fr.org	fr	20.63%	89.79%	33.56%	0.33
etudes-litteraires.com	fr	95.75%	84.31%	89.67%	0.1
ubuntu-gr.org	gr	85.81 %	62.18%	72.11%	0.39
dotnetzone.gr	gr	72.98%	90.72%	80.89%	0.37
fe-mail.gr	gr	1.80%	96.28%	3.54%	3.2
englishforums.com	en	71.31%	76,99%	74.04%	0.1
forum.ency-education.com	ar	94.46%	46.41%	62.24%	5.57
Average values		61.53%	75.03%	52.50%	1.61

TABLE XI
BOOTCAT

Domain	lang.	precision	recall	F-meas.	time
developpez.com	fr	36.15%	100%	53.10%	1.30
ubuntu-fr.org	fr	6.25%	99.75%	11.76%	0.6
etudes-litteraires.com	fr	81.85%	98.83%	89.54%	0.23
ubuntu-gr.org	gr	24.15%	98.02%	38.75%	1
dotnetzone.gr	gr	40.44%	96.61%	57.01%	0.55
fe-mail.gr	gr	16.13%	100%	27.77%	2.6
englishforums.com	en	23.99%	83.24%	37.25%	0.24
forum.ency-education.com	ar	64.91%	98.86%	78.37%	0.32
Average values		36.73%	96.91%	49.19%	0.85

VI. CONCLUSION AND PERSPECTIVES

We have presented DyCorC, an unsupervised system for extracting noiseless content from web forums, with an algorithm that does not use knowledge about language or tags. It outperforms the three state of the art models we tried on our gold standard corpus in four languages, in precision and F-measure, though it is outperformed by BootCat in computing time, taking 20% more time, but being still in a reasonable range.

TABLE XII
JUSTEXT

Domain	lang.	precision	recall	F-meas.	time
developpez.com	fr	34%	100%	50,75%	0,15
ubuntu-fr.org	fr	11,53%	100%	20,68%	0,1
etudes-litteraires.com	fr	82,23%	100%	90,25%	0,1
ubuntu-gr.org	gr	31,21%	100%	47,57%	0,15
dotnetzone.gr	gr	36,69%	100%	53,68%	0,11
fe-mail.gr	gr	55,11%	100%	71,06%	2,1
englishforums.com	en	44,41%	100%	61,50%	0,4
forum.ency-education.com	ar	54,91%	100%	70,89%	1,11
Average values		43,76%	100%	58,29%	0,52

Integration of DyCorC with BootCat could combine the efficacity of DyCorC as a boilerplate detector with the performance of BootCat as a crawler. But DyCorC also could be improved by being able to detect the best distance to be used, alternating Feature distance and Jaro distance, making an educated guess about the length of trees in a forum.

The problem of arabic forums slowing Nutch should be carefully assessed, as it still is one of the most interesting automatic crawlers (and, outside of this forum, the quickest). If it is due to BoilerPipe, replacing it with a library extracted from DyCorC could lead to even more interesting results than integrating it with BootCat.

REFERENCES

[1] J. M. Schulz, C. Womser-Hacker, and T. Mandl, "Multilingual corpus development for opinion mining," in *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, 17-23 May 2010, Valletta, Malta*, N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner, and D. Tapias, Eds. European Language Resources Association, 2010.

[2] J. Pustejovsky and A. Stubbs, *Natural Language Annotation for Machine Learning: A Guide to Corpus-Building for Applications*. O'Reilly Media, 2012.

[3] C. D. Lin Xu, "Approximate retrieval of xml data with approx- path," in *In proceedings of the nineteenth conference on Australasian database*, vol. Volume 75.

[4] D. Myers and J. W. McGuffee, "Choosing scrapy," *J. Comput. Sci. Coll.*, vol. 31, no. 1, pp. 83–89, 2015.

[5] C. Bosco, V. Patti, and A. Bolioli, "Developing corpora for sentiment analysis: The case of irony and senti-tut (extended abstract)," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, Q. Yang and M. Wooldridge, Eds. AAAI Press, 2015, p. 4188. [Online]. Available: http://ijcai.org/Abstract/15/587

[6] X. Li, K. Peterson, S. Grimes, and S. Strassel, "Bolt chinese-english word alignment and tagging - discussion forum training," *Linguistic Data Consortium*, 2016. [Online]. Available: https://catalog.ldc.upenn.edu/ldc2016t05

[7] C. Kohlschtter, P. Fankhauser, and W. Nejdl, "Boilerplate detection using shallow text features," in *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, ser. WSDM '10. New York, NY, USA: ACM, 2010, pp. 441–450. [Online]. Available: http://doi.acm.org/10.1145/1718487.1718542

[8] J. T. Mahara and K. Thirumoorthy, "A survey on web forum crawling techniques," *International Journal of Innovative Research in Science, Engineering and Technology*, vol. 3, March 2014.

[9] J. Jingtian, Y. Nenghai, and L. Chin-Yew, "Focus: Learning to crawl web forums," in *Proceedings of the 21st International Conference on World Wide Web*, ser. WWW '12 Companion. New York, NY, USA: ACM, 2012, pp. 33–42.

[10] A. J. McMinn, Y. Moshfeghi, and J. J. M., "Building a large-scale corpus for evaluating event detection on twitter," in *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, ser. CIKM '13. New York, NY, USA: ACM, 2013, pp. 409–418.

[11] M. Dref and A. Pappa, "An interaction approach between services for extracting relevant data from tweets corpora," in *CILC2016. 8th International Conference on Corpus Linguistics*, ser. EPiC Series in Language and Linguistics, A. M. Ortiz and C. P\'erez-Hern\'andez, Eds., vol. 1. EasyChair, 2016, pp. 97–110.

[12] L. Medrouk, A. Pappa, and J. Hallou, "Review web pages collector tool for thematic corpus creation," in *CILC2016. 8th International Conference on Corpus Linguistics*, ser. EPiC Series in Language and Linguistics, A. M. Ortiz and C. P\'erez-Hern\'andez, Eds., vol. 1. EasyChair, 2016, pp. 274–282.

[13] Y. Zhai and B. Liu, "Extracting web data using instance-based learning," in *WISE-05*. WISE, 2005, pp. 318–331.

[14] R. Khare, C. D., S. K., and A. Rifkin, "Nutch: A Flexible and Scalable Open-Source Web Search Engine," in *CommerceNet*, ser. CN-TR-04-04, Nov. 2005.

[15] M. Baroni and S. Bernardini, "Bootcat: Bootstrapping corpora and terms from the web," in *In Proceedings of LREC 2004*, 2004, pp. 1313–1316.