Crawling the Dutch Web

Yannick Hogewind*
yannick.hogewind@student.ru.nl
Radboud University
Nijmegen, Gelderland, The Netherlands

Kim Nendels*
k.nendels@student.ru.nl
Radboud University
Nijmegen, Gelderland, The Netherlands

Nienke Wessel*
n.wessel@student.ru.nl
Radboud University
Nijmegen, Gelderland, The Netherlands

ABSTRACT

Some institutions have a specific goal when preserving the world wide web. The Dutch Royal Library has been exclusively archiving the Dutch (including variants of the Dutch language) part of the web. Considering only one percent (approximately) of the web is in Dutch, this is a somewhat more complicated task, because standard methods, such as language detection based on content, would prove cumbersome and ineffective. In this paper we explore the method of language detection based only on URLs, using words and trigrams as features. We employ our method on the web by doing several example crawls 'in the wild', showing that our methodology increases the relative crawl yield with a reduction in completeness.

KEYWORDS

URL language detection, web crawling, Scrapy, trigrams,

ACM Reference Format:
Yannick Hogewind, Kim Nendels, and Nienke Wessel. 2020. Crawling the Dutch Web. In *Information Retrieval Project Report, December 21, 2020, Nijmegen, NL.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

In recent years, interest has spiked in archiving and preserving parts of the world wide web with the purpose of, for example, enabling future research into language use or into cultural aspects. While the majority of focus has been put on archiving the whole web, some institutions have a more specific segment in mind. An example is the Dutch Royal Library (Koninklijke Bibliotheek), which has been exclusively archiving the part of the web that is in Dutch, including variants of the Dutch language e.g. Flemish and Surinamese Dutch (*Surinaams*) [1]. Their goal, as stated on the website, is to archive all Dutch publications, including all publications available on the internet as well.

This poses an intricate task, for only an estimated 1% of the internet is in Dutch. Consequently, achieving this task would be ineffective if standard methods are used. Downloading every page and checking whether it is in Dutch or any variant will be too arduous. Instead, more creative solutions have to be found to tackle this problem.

Therefore, in our project, we will be using the ideas of language locality and language classification to find a method to crawl the world wide web for Dutch pages in a more efficient way. Our research question is "how can we efficiently obtain as many different Dutch web pages as possible?".

To answer this question, we will first look at work that looked at similar questions, then explain our approach, and finally discuss our results.

2 RELATED WORK

Most of the work on specific language web crawling has been done on the basis of either URLs, or specific search engines who have already done the majority of the work for you. Baykan et al. [4] provide a review of techniques for URL based language identification. Most interestingly, Baykan et al. [3] provide a rather successful way to crawl different Indo-European languages.

A non-URL but content-based approach was proposed by Remus and Biemann [7]. They use a predefined domain in which they are interested, and then calculate for every page encountered how well it fits this predefined domain. Then, web locality is exploited by exploring links on pages that score high on the domain match first. Similarly, Azimzadeh et al. [2] also base their approach on a suitability score which mostly depends on the page that was linked from, but focus mostly on the language of that page, and not on other properties.

Methods that depend on previous crawls include the method used by Tamura et al. of using premade language graph of the web to determine where to crawl [8]. Ghani et al. propose a method of crawling minority language websites online that involves crawling web pages and using the crawled web pages to generate language-specific search queries that can be used in a search engine to find more pages in that language [5].

3 METHOD

The source code used in this paper is available at https://gitlab.science.ru.nl/ghogewind/language-crawler.

3.1 Approach

Since only one percent of the web is estimated to be in the Dutch language or variants of Dutch, we have explored different approaches to web-crawling besides a conventional content-based language detection crawling approach. As mentioned before, apart from our selected methods of crawling, we exploit the assumption of language locality. This assumption purports that web pages written

*All three authors contributed equally to this research.
[1] https://www.kb.nl/organisatie/onderzoek-expertise/e-depot-duurzame-opslag/webarchivering

IR Project, December 21, 2020, Nijmegen, NL

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

in a certain language tend to cite other web pages in the same language. Also, these web pages are likely to be located near other web pages written in the same language. In practice, this means that our crawler will stop pursuing its current path if it finds a web page in a language other than Dutch. To this end, our main approach uses a language identification method based on the URL of a web page, building on work by Baykan et al. [3]. We train a Naive Bayes classifier that, given a URL, predicts whether the associated page is Dutch and apply this classifier in our crawler by only crawling URLs that this classifier considers to be Dutch. This approach has several strengths. Primarily, URL classification has been shown to produce good results in focused crawling and has a low memory footprint [4]. In addition, this approach does not rely on extensive crawls created by third parties.

3.1.1 Features. In order to classify URLs as Dutch or non-Dutch, we extracted different features from the URLs. These can be divided into two main sets: words and character 3-grams. These have been shown to work well on both shorter and longer texts [6]. Baykan et al. suggest using this for URL classification.

Word features are extracted by splitting each URL network location into a sequence of strings of letters at each non-letter character. Also, the path is separated at each /, and all these parts of the path are also considered as words. URL parts like "www" are disregarded as they were considered to provide little information on the language of the page. We used the 'words' obtained from the URL as such and took no further effort to separate them if they were compounds. That means that for a website like `persoonlijkrooster.ru.nl` we included `persoonlijkrooster`, `ru` and `nl` as words. We did not try to separate the first into `persoonlijk` and `rooster`, as that would require extensive knowledge of the language at play, for example by using a dictionary.

For the second set of features, the words from the first set were used to generate character trigrams. For the Dutch word "gezellig", we will end up with following trigrams: "_ge", "gez", "eze", "zel", "ell", "lli", "lig" and "ig_". The advantage of using trigrams over using words as tokens (as described in the method above), is that an algorithm can likely detect a language by learning which trigrams are common in a certain language. In this case, the trigram "ig_" is quite common in Dutch language. This can in turn be applied to unknown full word tokens to estimate whether its language is Dutch.

Baykan et al. [3] assume that creating tokens through the words-as-features-method and subsequently splitting the words in trigrams is less random than using the trigrams-as-features-method as a stand-alone method, and therefore preferred. We follow this suggestion.

One of the highlights of the study of Baykan et al. is that using words as features battles one of the tougher problems in detecting languages in URLs only: English is considered to be the "technical language" of the web. This means that English words are also often used in URLs for non-English web pages. By using words as features, the machine learning algorithm is able to recognize certain host names and as such correctly determine the language of an URL. However, this storing of previous knowledge only works if the training set of URLs is large enough. If the training data is reduced, trigrams perform better than words as features (by a factor of 10 or more, [3]).

For this reason, we chose to implement both the words as features and trigrams as features in our approach. This should theoretically yield good results based solely on language detection in URLs.

3.1.2 Classification algorithm. Baykan et al. [3] compared the success of these features with six different classification algorithms: Country code top-level domain only, Country code top-level domain plus.com and .org, Naive Bayes (NB), Decision Trees, Relative Entropy and Maximum Entropy. The first two algorithms, ccTLD and ccTLD+ only checked the country code top-level domain and assigned it to the language it corresponded to (e.g. .fr to French, .de to Germany). ccTLD+ in addition labeled .com and .org as well. In their results, they showed that using the individual classifier of Naive Bayes with words as features gave the best performance, with an F-score of .91 averaging over three separate tests by detecting five different languages (English, German, French, Spanish and Italian).

We therefore chose to implement the ccTLD baseline algorithm (checking for .nl domain names), along with the Naive Bayes classifier for our project. This means that the classifier first checks whether a URL has .nl as top level domain. If so, it crawls the page. If not, it uses naive Bayes classifier to guess the language of the page.

3.2 Implementation and experimental setup

The foundation of our web crawler depends on Scrapy. Scrapy is an open source and collaborative framework for extracting data from websites. We focused on extending Scrapy in such a way that it will be effective in language-specific crawling. In order to do this, we trained the naive Bayes classifier on an initial crawl containing 46 thousand pages.

This initial crawl was made by first doing a crawl without any restrictions, and then selecting a maximum of five URLs from each domain as to prevent the classifier from only learning specific domains. We then determined the language of the page from the first 500 bytes of rendered plain text. For language detection, we used the Python library `polyglot`[2]. This uses sequences of four letters (quadgram) with the naive Bayes classifier to determine the language of a text with high accuracy [1].

We then, for each URL in our training set, determined its language and made this a binary feature: 0 means non-Dutch and 1 Dutch. We then used this to train our multinomial Naive Bayes classifier[3].

3.2.1 Evaluation. The performance of the URL classifier was tested by calculating an F-score over the training set, using 80% for training and reserving the remaining 20% to test the performance.

Because we also wanted to know the effects of using a method like this in crawling, we also did crawls on the web. Papers like the paper of Baykan et al. only test their methods in artificial situations, where they only look at how well their classifier works on a test set, but do not employ it in making an actual crawl. We determined the efficacy of our crawling approach by performing four crawls on the

[2]https://pypi.org/project/polyglot/
[3]https://scikit-learn.org/stable/modules/naive_bayes.html

Seed	Share of Dutch pages in rejected URLs (95% CI)	n samples	n rejected URLs
Mixed	6.40% (± 2.15%)	500	7.464.226
Dutch	34.9% (± 4.09%)	522	704.171

Table 1: Share of Dutch pages in the rejected URLs for the URL filtered crawls, calculated over a sample of rejected URLs.

general web using different strategies, computing metrics over the generated corpora. Each crawl was set to either crawl all URLs it finds or to only crawl URLs our classifier considers to be Dutch. We also tried starting with two different seed lists: either only Dutch URLs, or an equal mix of Dutch and non-Dutch, European seed URLs. These crawls each ran for the same amount of wall-clock time. During the crawls the crawled pages were stored so that their true language could be identified afterwards; the two crawls that used our URL classifier also registered the URLs they rejected. Additionally, we tracked the amount of pages crawled over time.

After 4.5 hours, the crawls were stopped and metrics calculated. These metrics included the size of the corpus in number of pages and the number and percentage of the Dutch pages in the corpus, called "crawl yield" by Baykan et al. [3]. The ground truth language of the pages was determined using the same method by which we created the training set for our classifier, using `polyglot`. Finally, we calculated the percentage of URLs rejected by the classifier that were Dutch, which are false negatives. Since it would be infeasible to crawl the numerous rejected URLs to determine the true language of the pages, we sampled a random subset of the rejected URLs. We then determined the share of this sample that was Dutch, again using `polyglot`.

4 CONCLUSION

4.1 Results

The F-measure result achieved by our URL classifier is .918. We find this to be acceptable for our purposes, as the best performing classifier presented by Baykan et al. [3] resulted in an F-score of .91 average over all three test sets.

The metrics calculated over the four different crawl strategies are available in Table 2. We can see that the portion of Dutch pages, i.e. the crawl yield, is significantly improved when the URL classification filter is introduced since fewer non-Dutch URLs are crawled. This effect is especially pronounced when applied to the seed URLs containing a mix of Dutch and non-Dutch websites. In that case, the crawl without URL filtering crawls all pages, regardless of their language and therefore the crawl yield approaches the share of Dutch web pages on the internet. In the crawls seeded with Dutch pages only, the difference is smaller, but still significant. Here the unfiltered crawl is aided by the effect of language locality: Dutch pages often contain links to other Dutch pages.

While the yield percentage is larger for the filtered crawls, the total size of the corpus is smaller in both cases than the unfiltered crawls. This is most likely caused by the fact that the filters reduce the number of URLs added to the crawl queue, temporarily starving the crawler of URLs. An alternative explanation could be that applying the filters requires more time to process each page and slows down the crawl; however, this is unlikely since the filter application cost is on the order of milliseconds. In all cases, the crawl speed is unlikely to be limited by the network bandwidth available to our test machine since the crawl request rate and bandwidth use were far lower than the capabilities shown by preliminary tests.

A consequence of the lower corpus size for the filtered crawls is that the filtered Dutch seed crawl has a lower absolute Dutch page yield, even though its yield percentage is larger. The statistics for the rejected URLs in the two URL filtering crawls can be seen in in Table 1. The share of Dutch pages in the rejected URLs can be interpreted as the false negative percentage of our URL classifier over all URLs encountered in the crawl. These numbers are satisfactory: the mixed crawl had a share of 6.4% Dutch pages in the rejected URLs, with 33.6% Dutch pages in the corpus and the Dutch seeded crawl had a share of 34.9% Dutch pages in rejected URLs, with 67% Dutch pages in the corpus. In both crawls, the URL classifier performed much better than chance. We can conclude that our URL classification-based crawling increases corpus purity, sacrificing completeness and pages crawled per time unit.

4.2 Discussion

4.2.1 Limitations. Our first intention was to crawl using Apache Nutch combined with search component Apache Solr for language detection. This however proved too difficult because of lacking documentation and limited time. We therefore switched to an easier to use library that claimed to be as good as Nutch. This was Scrapy, using Python, with the CLD2 plugin. However, it turned out that Scrapy was not designed for large crawls and had some bugs, mostly that it turned out to be difficult to make Scrapy scrape in a different order than lexicographic. The problems with Nutch and Scrapy cost us a lot of time, leaving us with little time for the rest of the report.

Another limitation is that our calculation of the F-score is slightly biased. This is because some urls of the same domain (but with different paths) are in both the train and the test set. This was done because of time limitations, but means the F score needs to be interpreted with some caution.

4.3 Future research

Compared to similar papers, we had a relatively small training set of only 46 thousand pages (Baykan, for example, had at least 99k pages for every language). At least for separate word and 3gram features, Baykan et al. seems to show improvement in F-score with a larger data set. Therefore, making a bigger training set might be helpful. This might be aggravated in our case because our training set contains multiple URLs from the same domain, only differing in the path. Performance might increase if more diverse URLs are used.

As interest in varieties other than the standard variety is growing, more focus on nonstandard languages seems warranted. We paid some attention to it by testing how the polyglot library works with these varieties [4]. However, as it is quite hard to automatically

[4]We put some Surinamese and Flemish texts and list of words unique to that variant in the classifier. See https://www.reddit.com/r/belgium/ and https://www.reddit.com/r/Suriname/ for texts, and https://www.vlaanderen.be/taaladvies/standaardtaal-verschillen-tussen-belgi%C3%AB-en-nederland and

Seeds	URL filter	Total pages in corpus	Dutch pages in corpus	Portion of Dutch pages in corpus
Mixed	Not used	137,225	476	0.346%
Mixed	Used	79,519	26,715	33.6%
Dutch	Not used	123,824	62,684	50.6%
Dutch	Used	80,960	54,234	67.0%

Table 2: Metrics for each completed crawl.

determine of a Dutch text to which variety of Dutch it belongs, we could not automatically determine whether the texts we crawled were of varieties other than the standard one. Therefore, for future research, it seems useful to look into ways to automatically determine which variety of a language we are dealing with, and then use that to see how language specific crawling methods known thus far do on crawling non-standard varieties.

Furthermore, we feel that our method of testing the crawler on the real web, could be further improved and used to evaluate other crawlers. Currently, there is a tendency to test these kinds of classifiers only on training and test sets, but, of course, the ultimate goal is to apply them to the real world, and our evaluation method provides a way to see how they do there in more realistic situations.

REFERENCES

[1] Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. *arXiv preprint arXiv:1307.1662* (2013).

[2] M. Azimzadeh, A. Yari, and M. J. Kargar. 2010. Language specific crawling based on web pages features. In *2010 International Conference on Multimedia Computing and Information Technology (MCIT)*. 17–20. https://doi.org/10.1109/MCIT.2010.5444865

[3] Eda Baykan, Monika Henzinger, and Ingmar Weber. 2008. Web page language identification based on URLs. *Proceedings of the VLDB Endowment* 1, 1 (2008), 176–187.

[4] Eda Baykan, Monika Henzinger, and Ingmar Weber. 2013. A Comprehensive Study of Techniques for URL-Based Web Page Language Classification. (Mar 2013). https://doi.org/10.1145/2435215.2435218

[5] Rayid Ghani, Rosie Jones, and Dunja Mladenić. 2001. Mining the web to create minority language corpora. In *Proceedings of the tenth international conference on Information and knowledge management (CIKM '01)*. Association for Computing Machinery, 279–286. https://doi.org/10.1145/502585.502633

[6] Thomas Gottron and Nedim Lipka. 2010. A comparison of language identification approaches on short, query-style texts. In *European Conference on Information Retrieval*. Springer, 611–614.

[7] Steffen Remus and Chris Biemann. 2016. Domain-Specific Corpus Expansion with Focused Webcrawling. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. European Language Resources Association (ELRA), 3607–3611. https://www.aclweb.org/anthology/L16-1572

[8] Takayuki Tamura, Kulwadee Somboonviwat, and Masaru Kitsuregawa. 2007. A method for language-specific Web crawling and its evaluation. *Systems and Computers in Japan* 38, 2 (2007), 10–20.

http://nl.wikipedia.org/wiki/Lijst_van_verschillen_tussen_het_Nederlands_in_Belgi%C3%AB,_Nederland_en_Suriname for words not present in standard Dutch. Each time, the classifier still classified it as Dutch.