Bachelor Thesis

Identification and Information Extraction on Scientists Home Pages in the Common Crawl Web Archive

---

Samuel Roth

Examiner: Prof. Dr. Hannah Bast
Advisers: Prof. Dr. Hannah Bast

Albert-Ludwigs-University Freiburg
Faculty of Engineering
Department of Computer Science
Chair of Algorithms and Data Structures

October 05th, xxxx **(TODO:)**

Writing period

05. 07. xxxx – 05. 10. xxxx **(TODO:)**

Examiner

Prof. Dr. Hannah Bast

Advisers

Prof. Dr. Hannah Bast

Declaration

I hereby declare, that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.
I hereby also declare, that my Thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

Place, Date	Samuel Roth

Abstract

Aim of this work is the development of a system to detect personal Web pages of scientists and extract structured data - the scientist name, affiliation, gender, profession - from these personal Web pages. In addition we detect occurrences of the scientists in other Web pages texts to use these texts as information sources for search queries about the scientist in question. As a source of raw data we use *Common Crawl*[1] - a open archive of 3.16 billion+ crawled Web Pages. For the classification task we reviewed recent work on categorizing Web pages and implemented web-specific features to train a supervised binary classifier using existing and manually labeled data. To extract the correct name we utilized Named Entity recognition techniques and SVMs with special features for gender, profession and affiliation combined with available lookup tables to improve performance. Finally we use the retrieved data to create a Broccoli[1] index so we can use the data to search for existing scientists in different contexts.

[1]Common Crawl: `http://commoncrawl.org/`, [Online: Last accessed on 09.07.2017]

Zusammenfassung

In dieser Arbeit wird ein System entwickelt um persönliche Webseiten von Wissenschaftlern zu erkennen und aus diesen Informationen wie Namen, Geschlecht, berufliche Position und zugehörige Institution zu erhalten. Mit den erhaltenen Entitäten werden die Texte der übrigen Webseiten nach Vorkommen der Entitäten durchsucht und diesen zugewiesen. Mit diesen Daten wird abschließend ein Index für die semantische Suchmaschine Broccoli[1] erstellt. Als Dantenquelle verwenden wir das *Common Crawl*[2] Webarchiv. Für die Klassifizierung der Webseiten evaluieren wir bestehende Ansätze, implementieren aufgabenspezifische Features für die Webseiten und trainieren eine SVM mit bestehenden und selbst erstellten Datensätzen. Zur Eigennamenerkennung verwenden wir bestehende Algorithmen. Zur Bestimmung von Geschlecht, Position und zugehöriger Institution entwickeln wir eigene Features für Lineare Klassifikationsmethoden und kombinieren diese mit Lookup-Tabellen für ein verbessertes Ergebnis.

[2]Common Crawl: `http://commoncrawl.org/`, [Online: Last accessed on 09.07.2017]

Contents

1. Introduction

1.1. Motivation

The Web is the biggest collection of information humans ever created but inherent unstructured. The sheer size of the Web and the lack of structure makes it hard to handle, especially if one is looking for specific information. Classification of Webpages and the extraction of structured information are key tasks for mining data in the Web. For example most researchers have one or even many personal Webpages (*Homepages*). Hence structured data about a researchers name, profession, affiliation, research topics etc. is potentially available in the Web. Automatically mining structured data from the Web is a versatile and nontrivial task.

1.2. Problem Description

We try to identify personal Web pages of scientists and extract structured content from the pages to use them in the semantic search engine *Broccoli*[1]. To get from raw HTML data to a Broccoli index we face the following tasks:

- Fetching raw HTML data.
- Categorize Webpages into personal and non-personal pages.
- Find and extract the persons name the personal Webpage is about.
- Predict the persons profession, affiliation, gender.
- Find occurrences of the person entity in other thexts from a web corpora to use these as sources for semantic search.
- Parse the data into an ontology database and context documents as required by Broccoli.

1.3. System overview

We start by downloading all crawled Web pages from the Common Crawl archive for every university given by a list of domains. We remove non relevant parts called *boilerplate* from the HTML files and store the left over part for further processing. We create labeled training sets to be used in supervised classifiers and merge them with existing labeled data. We review related work on web page classification and decide for task specific features to be used in the classification task. On personal Web pages we apply named entity recognition algorithms to extract names on the page and predict the correct name - the one the page is about - using position features of occurrences for the different names. Then we utilize SVMs using text features to predict the profession and SVMs using text features combined with lookup-tables for gender detection. Having the scientist entities we detect their occurrences in the non-personal Web pages and store the matched texts as a source for semantic search documents. Finally we build a index for the semantic search engine Broccoli using the gathered facts on scientists to build an ontology database and the text documents to build the context corpora.

2. Related Work

Classification of web pages and in particular identifying personal pages is a recurring problem when building digital person related search engines such as aminer[1], dblp[2], Google Scholar[3] amoung a huge number[4] of applikations alike.

Approaching the Web page classification problem Qi et al.[2] classifies the task as a type of supervised learning problem and subdivides it into categories: Subject classification, functional classification, sentiment classification. Where functional classification cares about the role of a webpage, if it is a "personal homepage", "course page" etc. which we focus on in this work.

Regarding features used for categorization Kan et al. [3] showed, that only using URL features yield good results on webpage categorization. Qi et al. proposed the combination of content and link based features. Das G. et al.[4] made suggestions for URL features in the task of researcher homepage classification and showed, that they provide good additional evidence for homepage classification which is used in this work. Support Vector Machines (SVM) tend to be well suited for all kinds of categorization tasks (Joachims [5]) and are therefore used in this work for classification.

[1] `https://aminer.org/`
[2] `http://dblp.uni-trier.de`
[3] `https://scholar.google.de/`
[4] `https://en.wikipedia.org/wiki/List_of_academic_databases_and_search_engines`

3. Background and Implementation

We divided the process of getting from archived web crawling data to the structured data about scientists into four main steps as shown in Figure 1.

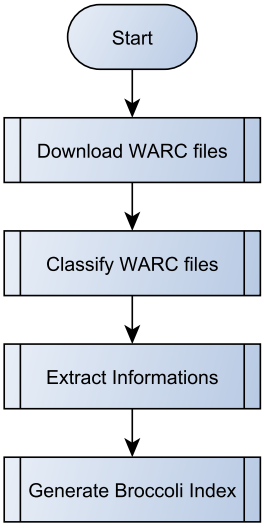

Figure 1.: Meta workflow of the approach used.

The next three sections of this chapter each describe the implementation for one of the steps with additional background information.

3.1. Download WARC files

Crawled Web pages in the Common Crawl archive are stored in the WARC[1] format which consists of a Header containing meta information about the content concatenated with the associated HTTP response in the body. In the Common Crawl archive a number of single WARC files are packed together into compressed files (called *WARC collections*). One complete crawl contains about 3 billion WARC files and is

[1] WARC - Format `https://en.wikipedia.org/wiki/Web_ARChive` [Online; last access 13.01.2018]

distributed into about 70.000 WARC collections resulting in about 43.000 WARC files per collection. Since the origins of the crawl results are random, Common Crawl provides a URL Index[2] to allow downloading an explicit URL by mapping the URL to the WARC collection and byte position the crawled data is located. The archive filename and position can be used to download a specific WARC record from the archive.

Because processing the whole Common Crawl archive would cost to much storage and computation time we used a collection of university domains to reduce the amount of processed data. The world-universities-csv [6] project was used to retrieve 9599 university domains.

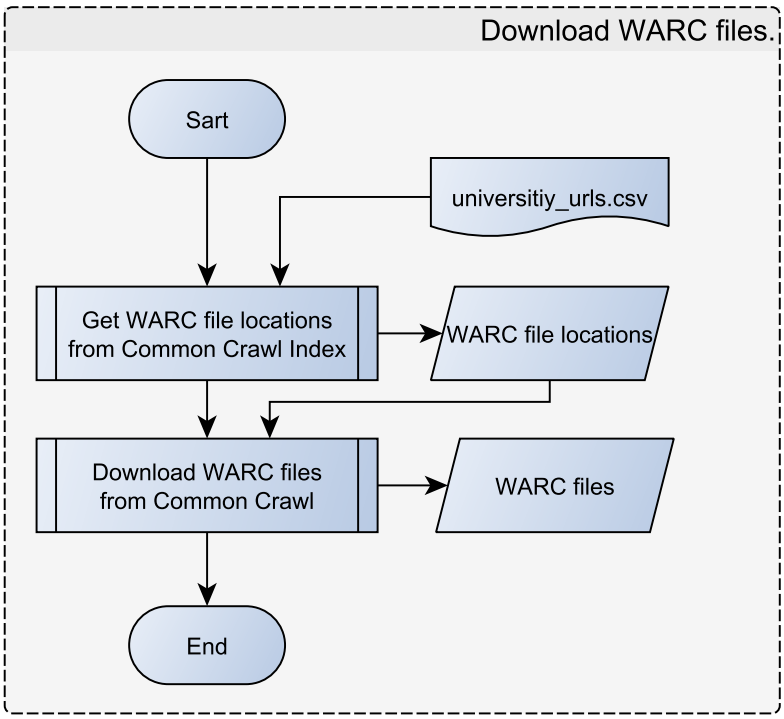

Figure 2.: Process of downloading WARC files.

Figure 2 provides an overview about the tasks done in the download process. For each university domain we extracted the positions of all websites covered by Common Crawl using the Common Crawl archive of the March 2017 crawl[3]. In total we

[2]Common Crawl Index `http://index.commoncrawl.org/` [online; last access 13.01.2018]
[3]March 2017 Crawl `http://commoncrawl.org/2017/04/march-2017-crawl-archive-now-available/` [Online; Last access 13.01.2018]

extracted 38.517.248 WARC file positions and downloaded the WARC content (about 400GB compressed data) for further processing.

3.2. Web page Classification

3.2.1. Classification process implementation

The classification process (figure 3) consists of 5 main tasks.

First we have to remove boilerplate, which means to extract relevant context from the Web page and dropping irrelevant content like navigation, copyright remarks etc. We continue by choosing and extracting features used as input for the classification algorithms, building a labeled training set, train the classifier and finally feeding the feature vectors of unseen sites into the classifier to classify these web pages.

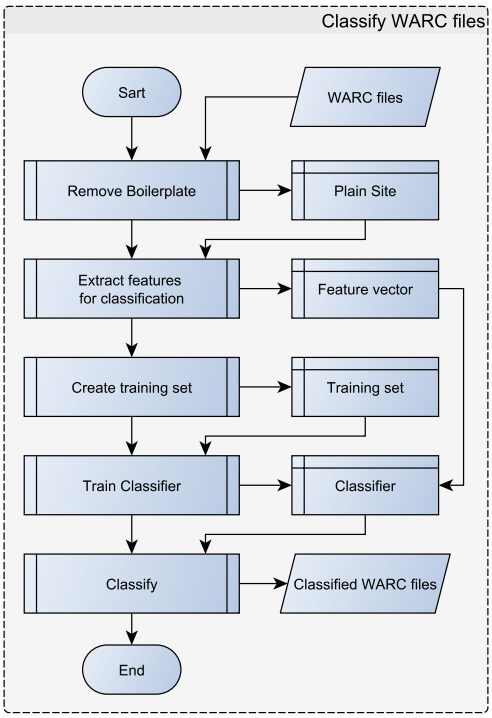

Figure 3.: Process of Web page classification.

3.2.1.1. Remove Boilerplate

Besides the relevant content, most web pages containing elements which are not part of the actual content also called *Boilerplate* [7]. This can be design elements,

navigation, header, footer or other additional elements as shown in Figure 4.

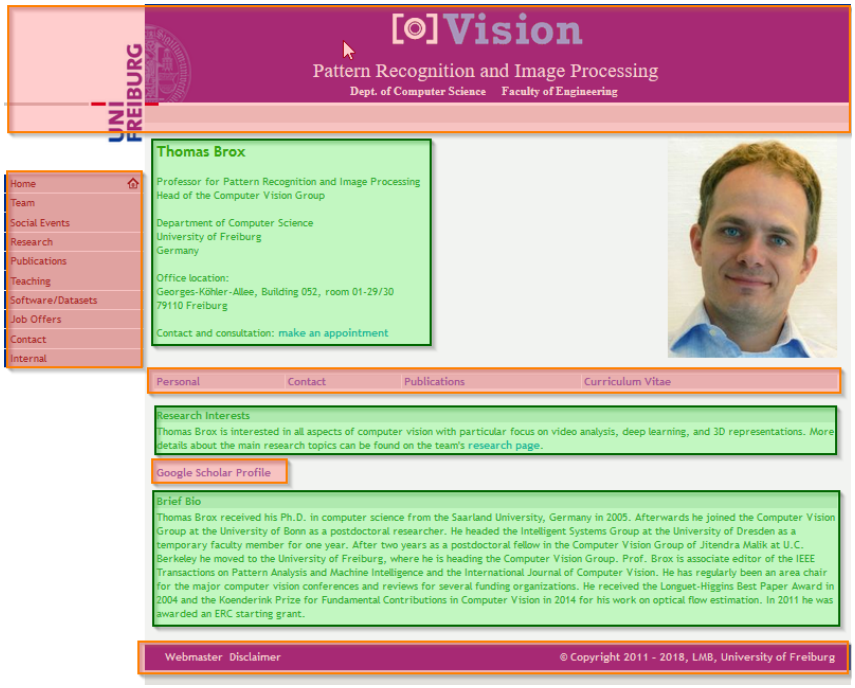

Figure 4.: Boilerplate in personal Web pages.
Relevant content in green boxes.

To extract relevant content we used the *jusText* [7] algorithm which is designed to preserve mainly text containing full sentences and it is therefore well suited for creating linguistic resources such as Web corpora[4]. After removing the Boilerplate we used the page *title* given by the string in between the `<title></title>` tags, *content text*, and *headings* returned by the *jusText* algorithm in the further process. Since it appears a feasible assumption, that content positions will gain information about relevance for certain content parts, we also kept track of the headings and text parts order by assigning integers in the order they occurred.

3.2.1.2. Choosing features

Web page classification relates strongly to text categorization which uses typically a "bag of words" approach [5] [8]. This means to collect terms frequently occurring in the training data to generate a dictionary. In addition we used used n-grams[5]

[4] jusText python implementation `https://github.com/miso-belica/jusText` [Online; last access 18.02.2018]
[5] https://en.wikipedia.org/wiki/N-gram [Online; lsat access 16.01.2017]

of words (after grid searching for the best performance as described in Section 4.2 we ended up using 1-, 2-, 3-word-grams). Word n-grams are able to capture the concept expressed by a sequence of terms [2]. Pages in our system are represented as normalized term frequency–inverse document frequency vectors[6].

Based on the observation, that URLs often contain strong features for webpage classification as shown in Table 1.

Nr.	URL
(1)	`https://www.inf.ethz.ch/personal/markusp/`
(2)	`http://www.cs.cmu.edu/~wpdann/index.html`
(3)	`https://ece.cmu.edu/news/article/439687/`
(4)	`http://www.informatik.uni-freiburg.de/institut/press`

Table 1.: Empirical URL features for webpage classificatioin.
Parts indicating personal pages in green. Red parts against.

Gollapalli et al. [4] suggested to also use URL features as additional evidence and have shown, that URL based features perform good especially on the researcher homepage identification task. We adapted the algorithm and used the version shown in Algorithm 1 which extracts URL parts and converts them to features. For example the Algorithm yields

`http://www.cs.cmu.edu/~wpdann/index.html`
$\Rightarrow$ `urlparamtildenodict, urlparamindex`

We padded the resulting URL features with `urlparam` and concatinated the features with the retrieved web page texts to transform them into a tf–idf vector. In addition we predicted that headings and n-grams of the headings will provide additional evidence for the categorization. The Idea originates from common patterns in researcher home pages which often include sections like "*CV*", "*Current Work*", "*Research interests*", "*Publications*" etc. Using n-grams will conserve more of the context since for example "*Publications*" is quite common to also appear in non personal pages but preceded by the heading "*CV*" it might be a strong feature in favor of personal pages. We prefixed the Headings using the string `headingparam` to be able to use this feature in the same tf–idf vector without inducing overlap to words in the actual context and therefore distort the contained information.

[6] tf–idf `https://en.wikipedia.org/wiki/Tf-idf` [Online; last access 14.01.2018]

Algorithm 1 Extract Url features

```
function getUrlFeatures(url)
    url_params ← getUrlParams(url)          ▷ delete extensions (.html, ...)
                                            ▷ Splits the URL by "/"
    result ← list()
    for param in url_params do
        if param starts with ~ then
            param.remove(~)
            if param is word then
                result += 'urlparamtilde' + param
            else
                result += 'urlparamtildenodict'
            end if
        else if param.length ≥ maxlen_param then
            result += 'urlparamlongword'
        else if any of (−, _) in param then
            result += 'urlparamhyphenatedword'
        else if param is digit then
            result += 'urlparamnumeric'
        else if param is word then
            result += 'urlparam' + param
        else
            result += 'urlparamnodict'
        end if
    end for
    return result
end function
```

3.2.1.3. Create training set

The training set we used consists of raw Html files associated with a label dictionary:

$$\begin{aligned} category &: \{personal \texttt{ or } nonpersonal\} \\ name &: \{\text{The scientists name } \texttt{or } None\} \\ gender &: \{\text{The scientists gender } \texttt{or } None\} \\ profession &: \{\text{The scientists profession } \texttt{or } None\} \end{aligned}$$

As a source for already labeled Html files we used the *WebKb*[7] dataset containing 2,902 personal pages of scientists and 5,214 negative examples. In addition we utilized the *dblp - computer science bibliography*,[8] which is a service that provides open bibliographic information on major computer science journals and proceedings. Authors covered in *dblp* may be associated with a personal page.[9] Using this information we extracted 1,916,530 additional personal pages. Investigating the retrieved pages from the *dblp* revealed that a lot of the pages where just profiles on social media services or landing pages of universities. Therefore *dbpl* can not be used out of the box and we filtered the URLs removing all common social media hosts and URLs pointing only to the top level of an institution. After filtering we downloaded the Html pages and were left with 21,991 additional personal pages. We then manually collected 246 personal pages for scientists at the University of Freiburg. For negative examples we randomly choose pages from the total set of available Html files downloaded from Common Crawl and manually filtered personal pages to be left with a set of 25,139 *nonpersonal* pages. In total our training set consists of 25,139 personal and 25,139 negative labeled pages.

For the remaining data extraction tasks we manually labeled set 296 names, 114 genders and 144 professions using a specific label software written for this task[10].

3.2.1.4. Train and classify

For the page classification we used the scikit-learn[11] implementation of a linear SVM. To prepare the SVM input we first concatenated the URL features, Heading features

[7] The 4 Universities Data Set `http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/` [Online; last access 19.02.2018]

[8] dblp - computer science bibliography `http://dblp.uni-trier.de/` [Online; last access 20.02.2018]

[9] Dblp: What is the meaning of these www-Home-Page records? `http://dblp.uni-trier.de/faq/What+is+the+meaning+of+these+www-Home-Page+records` [Online; last access 20.02.2018]

[10] PPE Labeler: `/ccp/code/flask_app`

[11] scikit-learn `http://scikit-learn.org/stable/index.html` [Online; last access 20.02.2018]

and page content text. Then we created a count vector for the 1-, 2- and 3-word-grams and used a maximum of the top 20,000 tokens while clamping the document frequency (equation 1) between 0.004 and 1. To normalize the term-document matrix we created a tf-idf term-document matrix using equation 4 to calculate the tf-idf values. The absolute values used where determined by the scikit-learn grid search implementation for hyper-parameter optimization.

$$df(term, docs) := |\{doc \in docs : term \in doc\}| \tag{1}$$

$$idf(term, docs) := log\left(\frac{|docs| + 1}{df(term, docs) + 1}\right) + 1 \tag{2}$$

$$tf(term, doc) := \frac{|\{token \in doc : token = term\}|}{|doc|} \tag{3}$$

$$tf - idf(token, doc, docs) := tf(term, doc) \cdot idf(term, docs) \tag{4}$$

To train the SVM we used 80% of the available training data and the remaining 20% to evaluate the classifiers performance. Results on performance are documented in the evaluation Section 4.2.

3.3. Extract data from personal pages

3.3.1. Retrieve the name

For retrieving the correct name of a personal page, we first extracted named entities appearing on the page using the Stanford Named Entity Recognizer (NER) [9]. After identifying person entities in the site content we face two problems when more then one name is extracted. First: Which name is the "correct" name and secondly which names represent the same person. These problems are illustrated in Figure 5.
As a solution we first detected and merged aliases for the same names occuring on a page into one name using Algorithm 2.
For the merged names we assigned each name a vector $v_{ner}(name)$ encoding the

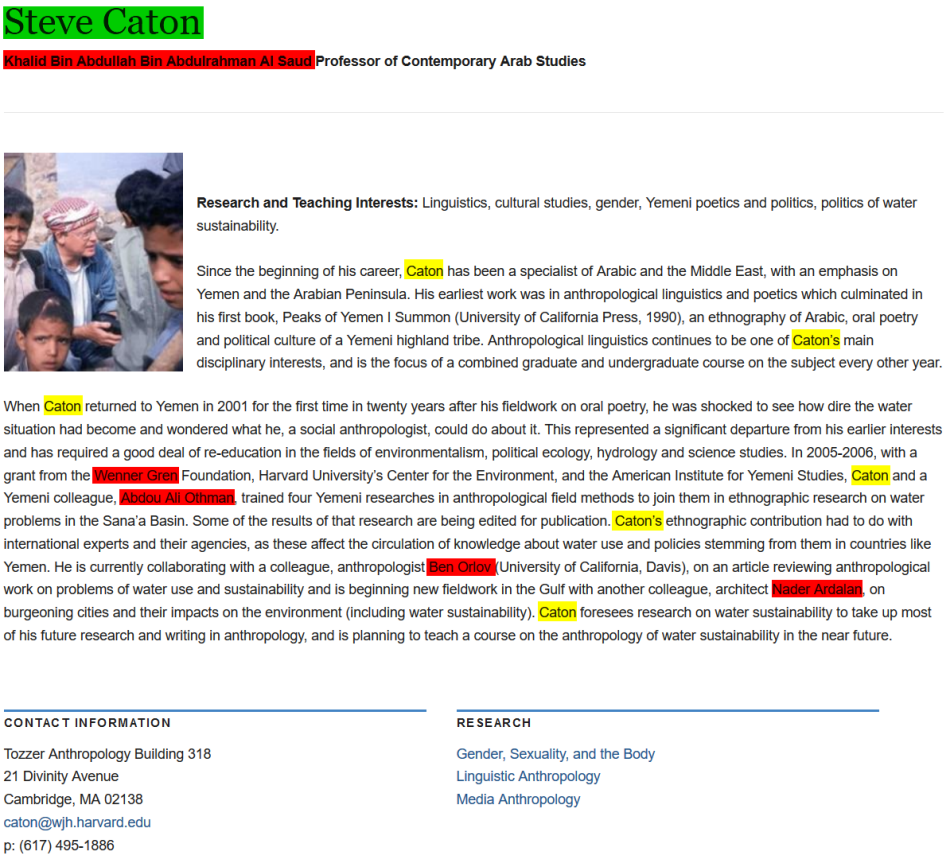

Figure 5.: Ambiguity and diversity in person names.
Associated name in green, aliases in yellow. Other names in red.

occurrences of *name* in the page content:

$$v_{ner}(name) := \begin{pmatrix} name \texttt{ in } title, \\ name \texttt{ in } heading^0, \\ \dots, \\ name \texttt{ in } heading^9, \\ name \texttt{ in } paragraph^0, \\ \dots, \\ name \texttt{ in } paragraph^9 \end{pmatrix} \tag{5}$$

To distinguish headings and paragraphs we reused the *JusText* algorithm results used for Boilerplate removal which distinguishes between text and headings and divides text into sentences. The vector v_{ner} was then used to train a SVM to yield the name

Algorithm 2 Find and merge alias names in a single page.

```
available_names ← getNamesUsingNer(page)
function nameMergeAlias(name)
    new_name ← name
    if name substring of any available_names then
        new_name ← matched name in available_names
    end if
    if new_name ≠ name then
        return nameMergeAlias(new_name)
    end if
    return new_name
end function
```

of the person the webpage is associated with.

3.3.2. Retrieve Profession, Gender and Association

For assigning an association to a retrieved scientist we mapped the domain host of the personal webpage URL to the educational institution given by the World Universities database [6].

To retrieve the profession we utilized the already available features used for page categorization but omitting the URL features since they do not provide significant evidence for the persons profession.

For determining the gender we used the pages content text 1-, 2-, 3-word-gram features with the same tf-idf representation used to categorize the pages (section 3.2.1.4) to train a SVM. In addition we collected name ⇒ gender mappings using different sources (List of the Sources available in Section A.1) yielding 97,220 names with their occurrences for male or female gender. To predict the actual gender we used Algorithm 3.

3.4. Build a broccoli Index

Building a broccoli index in our case consists of three main tasks (Figure 6).

First we build a knowledge base using our retrieved data to generate entries for each person as shown in figure 7.

Algorithm 3 Predict gender.

function StatisticGender(*name*)
  gender ← gender with more occurrences for *name*.
  $gender.probability \leftarrow \frac{\text{number gender was assigned for } name}{\text{total number } name \text{ occures in data}}$
  return gender
end function
function predictGender(*page*)
  name ← getName(*page*) ▷ As described in section 3.3.1
  gender ← *SVMPredict*(*page*) ▷ Using content text features.
  stat_gender ← StatisticGender(*name*)
  if *gender* ≠ *stat_gender* and *stat_gender.probability* > 0.7 **then**
    gender ← *statistic_gender*
  end if
  return *gender*
end function

```
Jane Doe is-a Person .
Jane Doe is-a PostDoc .
Jane Doe affiliation Brown University .
Jane Doe gender Female .
```

Figure 7.: Broccoli Knowledge Base (ontology) entry.

Next we built a collection of context texts called *postings* in Broccoli. This means we need to identify text associated with a person in the knowledge base. An obvious source is the personal page mapped to the scientist. In this page we used the retrieved content text sentences (as described in section 3.2.1.1). For this texts we decided whether or not to use it as a posting using algorithm 4.
As an additional source we identified all name entities in the content texts of all nonpersonal webpages available in the crawl. If an entity matched with the set of available persons - meaning an exact name match, we split the content into sentences and used the sentences containing the name as additional postings.

To score the persons in broccoli, which determines their order in search query results, we counted the occurrence of a person in other pages then their assigned personal pages. Each occurrence increased the score of a person by 1.

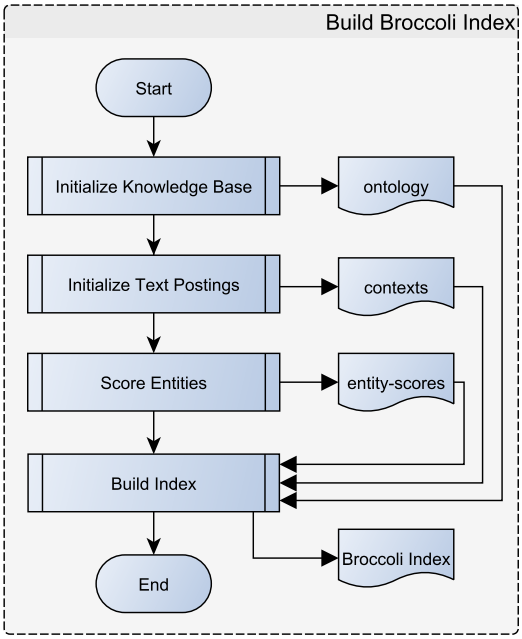

Figure 6.: Building a broccoli index flow chart.

Algorithm 4 Is Personal Text relevant.

function IsPersonalTextRelevant($person, text$)
  if $person.name$ **in** $text$ **then**
    return $true$
  end if
  if $person.is\_male$ **and** $male\_pronoun$ **in** $text$ **then**
    return $true$
  end if
  if $person.is\_female$ **and** $female\_pronoun$ **in** $text$ **then**
    return $true$
  end if
  if $[my, we, our]$ **in** $text$ **then**
    return $true$
  end if
  return $false$
end function

4. Evaluation

We are mainly interested in the quality of classification and data extraction. Additionally we measured crawl coverage in the Common Crawl archive to assess the quantities in our results and put the real world recall measurements into perspective.

We start by comparing the coverage of crawled pages in the availabe Common Crawl archives for a given set of scientists. In Section 4.2 we evaluate the performance of our classification models on the test data set and in Section 4.3 we evaluate the real world quality of the generated broccoli index by measuring results based on specific lists of scientists. In the end of this chapter in Section 4.4 we informally point out the main runtime factors and results for our implementation.

4.1. Coverage of Web pages in common crawl

As mentioned we used the March 2017 Common Crawl archive[1] as a source for raw data. Downloading all crawled webpages for the 9.599 domain hosts given by the 'World universities'[6] database yielded 38.517.248 webpages.

To measure the crawl coverage for our task we manually created a list of 334 personal home pages hosted by `http://uni-freiburg.de`. The coverage in the March 2017 crawl for this list is 28.74%. To judge on the coverage quality we compared this result to all available crawls. Figure 8 shows the results for all available archives from Summer 2013 till the June 2017 archive. We can see, that the coverage varies between 2.69% and 28.74% which means it would make sense to union different crawls to get a better coverage, which would result in a maximum coverage of 48.50%. In fact a portion of the URLs Common Crawl includes in a crawl are chosen randomly and based on *Harmonic Centrality*[2]. While measuring the recall of personal pages for different scientists in Section 4.3.1 we discovered a big difference in Common Crawl coverage between *Uni-Freiburg* hosted pages and a

[1] March 2017 Crawl `http://commoncrawl.org/2017/04/march-2017-crawl-archive-now-available/` [Online; Last access 13.01.2018]

[2] Hramonic Centrality; `https://en.wikipedia.org/wiki/Centrality#Harmonic_centrality` [Online; last Access 08.03.2018]

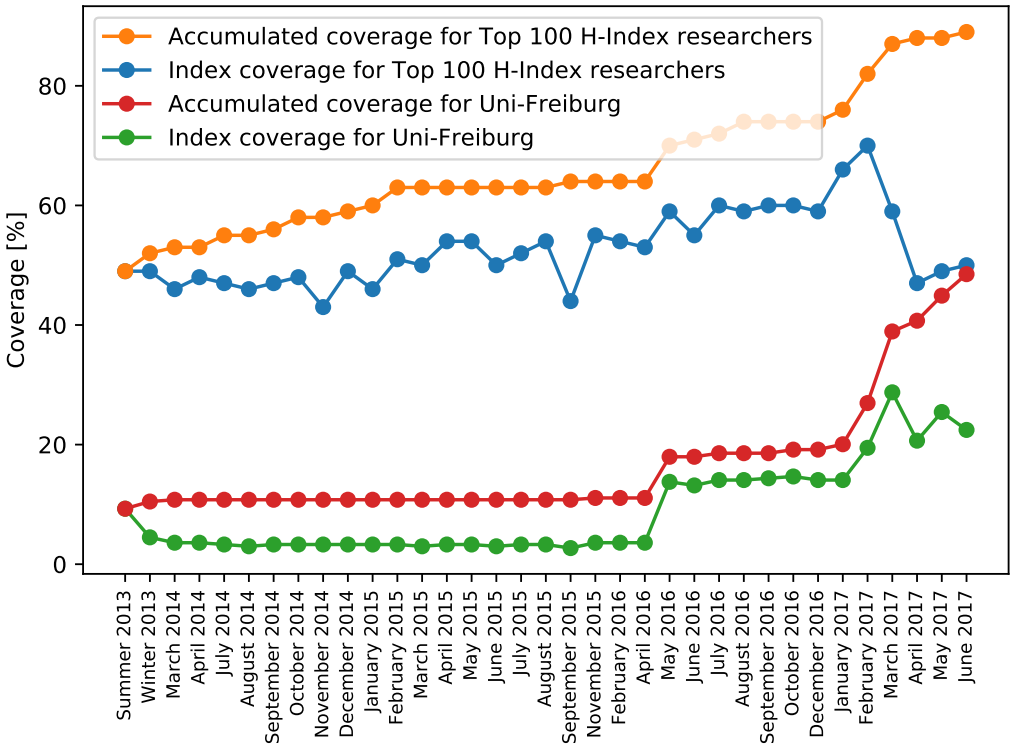

Figure 8.: Coverage personal webpages in the Common Crawl arcives.
334 Uni-freiburg personal pages and the top 100 researchers by h-Index given by `http://www.guide2research.com/scientists/`.
Uni-Freiburg: Max. accumulated 48.50%, max. single 28.74%
Top 100 h-Index: Max. accumulated 89%, max. single 70%.

list of the top 100 researchers given by google scholar h-Index[3]. Therefore we also measured the coverage and accumulated coverage in all Common Crawl indexes for the top 100 h-Index list. The better coverage for personal webpages of 59% in the March 2017 crawl and in general can be explained by the probably higher ranking of these pages (or pages near them in terms of similar URLs) due to their popularity. Common Crawl will therefore more likely crawl these pages as a part of the URLs Common Crawl includes in the archive is determined by a graph using links between pages as edges and then rank nodes by their degree. As a consequence webpages with more incoming links will be more likely considered to be taken into the archive.[4]

[3] h-Index; `https://en.wikipedia.org/wiki/H-index`; List of top 100 researcher by h-Index as listed by `http://www.guide2research.com/scientists/` [Online, last access 06.03.2018]

[4] Common Crawl Webgraph `http://commoncrawl.org/2018/02/webgraphs-nov-dec-2017-jan-2018/` [Online; Last access 08.03.2018]

4.2. Evaluating the performance of the classification models

The complete set of labeled data used to train and test our models consists of 50.248 webpages 25.139 labeled as *personal* and 25.139 labeled as *nonpersonal*, 296 personal webpages with the associated name, 114 personal webpages labeled with the associated gender and 144 personal webpages with the persons profession. To measure the performance of our classifier we randomly split our labeled data into two sets. The first sized 80% of available data was used to train the classifier and the remaining 20% were used as a test set to measure the classifiers performance. In order to get a more representative measurement we repeated the training 10 times with different random seeds and averaged the results. In addition we used the grid-search implementation of scikit-learn[5] for hyper parameter optimization of the classifiers. The final scores for each classifier are shown in Table 2.

Classifier	Precision $\frac{tp}{tp+fp}$	Recall $\frac{tp}{tp+fn}$	F_1-Score $\frac{2}{\frac{1}{Precision}+\frac{1}{Recall}}$	Test size
Page	0.89	0.89	0.89	1613
Name	0.90	0.88	0.89	46
Gender	0.81	0.74	0.75	19
Profession	0.58	0.59	0.47	22

Table 2.: Classifier performance measurements.
Values are weighted average over all labels. Test size is combined for all labels.

For the main task of categorizing the webpages into personal and non personal we reached a F_1-Score of 0.89. For a more detailed interpretation of the model learned by the page classifier, we extracted the 20 most important features shown in Figure 9. The most important feature in favor of a personal webpage is `:url:~nodict` which translates to webpages with URLs ending with a tilde followed by a word not in a dictionary. This is no surprise due to the historic use of the tilde indicating a home folder on Unix systems and in URLs the tilde mimics the Unix shell usage.[6] Also the headings "Research Interests" is a logical strong indicator for a personal page. On the other side for our model the heading "Faculty" turned out to be the strongest

[5] GridSearch hyper-parameter tuning `http://scikit-learn.org/stable/modules/grid_search.html` [Online; Last Access 11.02.2018]

[6] Tilde usage in URLs `https://en.wikipedia.org/wiki/Tilde#Directories_and_URLs` [Online; last access 11.03.2018]

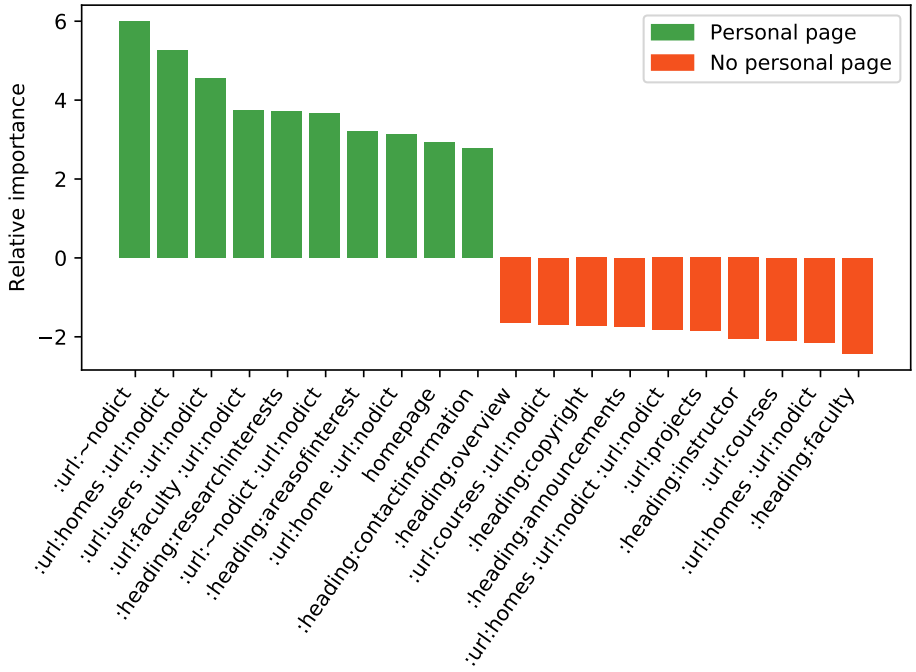

Figure 9.: Top 20 features for page classifier.
":url: nodict :url:nodict": Means `/~nodict/nodict` (Algorithm 1).
":heading:<text>": Is a heading with `<text>` in the page.

indicator against a personal page, which is not obvious and may indicate a bias in the training set. The relative importance of the features against personal pages are significantly lower than the features in favor of personal pages which models the lack of unambiguously features for non personal pages.

4.3. Evaluating Broccoli Index results

Generating a broccoli index resulted in a index of 39,017 persons - about one identified and successful processed personal page each 1000 webpages. On average each person is associated with 4.65 context documents which is a total of 181,258 texts with an average length of 24.73 words. The retrieved persons are distributed over 1,563 institutions where the top 10 universities yielded 35% and the top 50 66% of the results.

4.3.1. Recall for specific entities.

We collected and evaluated the recall for a number of persons as shown in Table 3.

Source	Total	Common Crawl		Broccoli Index		
	n_t	n_c	$p_c := \frac{n_c}{n_t}$	n_b	$p_{cb} := \frac{n_b}{n_c}$	$p_b := \frac{n_b}{n_t}$
`informatica-feminale.de`[7]	113	26	23,01%	5	19,23%	4,42%
`uni-freiburg.de`[8]	334	96	28,74%	21	21,88%	6,29%
`Top 100 H-Index researchers`[9]	100	59	59,00%	14	23,73%	14,00%

Table 3.: Recall for specific entities.
Full lists can be found in Section A.2.

The absolute numbers are significantly low, for example a recall of only $4,42\%$ for personal pages collected from the `informatica-feminale` list. A main reason is the low coverage of $23,01\%$ in common crawl. The recall of available sites in this case of $19,23\%$ can be explained by the structure and content of the given personal pages. For example the personal page URL

`http://www.informatik.uni-bremen.de/soteg/virthos.php?-pg=111`

Yields no positive features due to its generic form. Since our classifier is mainly text based, personal pages as shown in Figure 10 are likely to be categorized as *nonpersonal* since they do not provide enough classification features.

Another common issue is the excessive use of tables to structure the webpage as shown in Figure 11 which leads to fragmentation of the content and finally causes *jusText* [7] - the algorithm to retrieve relevant content - to classify the the webpage completely as boilerplate and therefore the page yields no classification features.

4.3.2. Quality of the results

To measure quality we randomly choose 185 index results given by the query

```
triples=$1 is-a :e:Person;$1 occurs-with robotics
```

[7] informatica-feminale.de `https://www.inforamatica-feminale.de/Professorinnen/Uni/listeuni.html` [Online; Last access 02.03.2018]
[8] Manually generated list of 334 personal pages hosted by `http://www.uni-freiburg.de`
[9] Manually generated list of top 100 H-Index researchers as listed by `http://www.guide2research.com/scientists/`

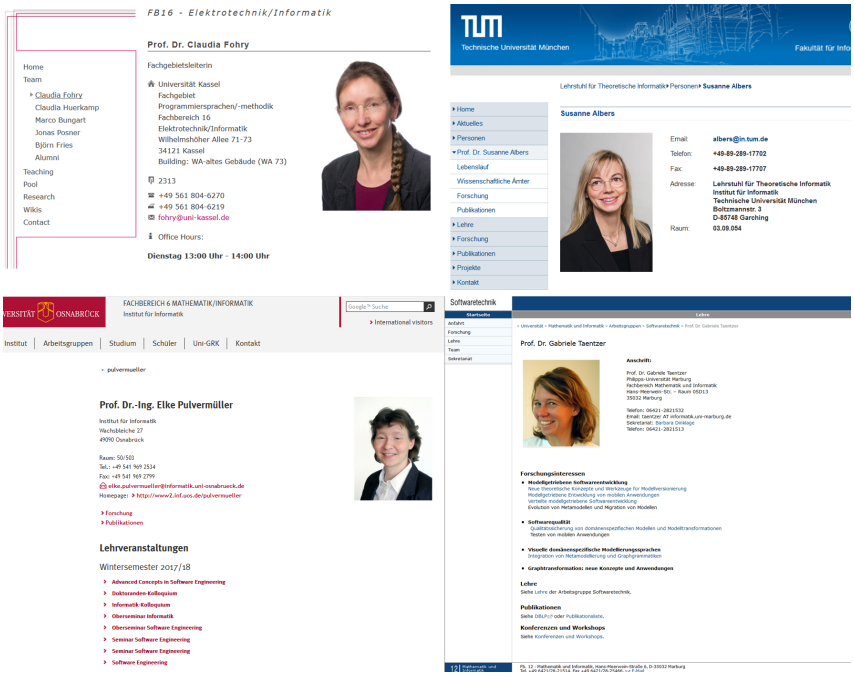

Figure 10.: Personal pages with meager text content.

which results in a list of persons occuring with the word *robotics* in the context texts. For this list we evaluated the quality of resulting names, if they are valid: For example '*Jason J. Corso*' or invalid ones like '*Pura Vida*' or '*Viterbi Voices*'. Then we examined if the linked webpages are in fact personal pages and if so, if the retrieved name is also the correct name of the person the page is about. Results are shown in Table 4.

Source	Total	Is name	Is personal page	Is correct name
Persons occurring with '*robotics*'	185	92.97%	79.46%	94.56%

Table 4.: Quality of Broccoli index.
Is name: The retrived name is a valid name. *Is personal page*: The linked page is a personal page. *Is correct name*: If linked to a personal page: Is the name the one the page is about.

The most interesting figure in Table 4 is that 79.46% of the resulting entries are originating from a personal page. This is a bit lower than what we expect given the precision value of 0.89 (Table 2) for our page classifier, which is a sign, that we slightly over-fitted our classifier.

Figure 11.: Excessive usage of tables in personal webpages.
On the left the rendered webpage. Right only the `<table>` structure used in this page. Source: `http://wwwiti.cs.uni-magdeburg.de/~jdittman/`

4.4. Runtime evaluation

All development and evaluation were done on a single machine with specifications as described in Table 5.

	Hardware Specifications
Processor	Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
RAM	16GB DDR3 1333 MHz
HDD	2 x Western Digital RE4 (3TB) using RAID 0
Internet connection	1 GBit/s synchron

Table 5.: Test system hardware specifications.

The most time consuming part of the system is downloading the WARC files from the Common Crawl archive which took about 12 days in total and then removing Boilerplate and extracting the text contents for the 38,517,248 WARC documents. Since we stored the results of every step separately to be able to re use the results in future improvements of the system, we produced a lot of overhead which could be avoided in a production system. In our implementation we used mainly multi-

processing where applicable and the bottleneck was mostly CPU power except for downloading the WARC files from the Common Crawl archive where the network speed was the limiting factor. A detailed list of runtime per task are provided in Table 6

Process	Task	n Tasks	Speed	Runtime	Bottleneck
Download WARC	Get locations	9599	$1.67\frac{loc}{s}$	$4.5h$	Internet Connection
	Download from S3	38,517,248	$37.65\frac{WARC}{s}$	$11.8d$	
Extract WARC	Get Html	38,517,248	$517.6\frac{WARC}{s}$	$20.7h$	CPU
	Rem. Boilerplate	38,517,248	$98.44\frac{WARC}{s}$	$4.5d$	CPU
Train	Train models	40.222		$74.7s$	CPU
Classify	Page	38,517,248	$448.39\frac{PPE}{s}$	$23.9h$	CPU
	NER	39,017	$126.45\frac{PPE}{s}$	$308s$	CPU
	Predict attributes	39,017	$452.2\frac{PPE}{s}$	$86s$	CPU
Broccoli Index	NER	38,478,231	$131.42\frac{Texts}{s}$	$3.4d$	CPU
	Generate KB	39,017	$452.2\frac{PPE}{s}$	$86s$	CPU
	Add texts	181,258	$71.9\frac{Text}{s}$	$0.7h$	CPU
	Score Entities	181,258	$43.6\frac{Text}{s}$	$1.1h$	CPU

Table 6.: Detailed runtime evaluation.

5. Conclusion and Future Work

We implemented a system which extracted 39,017 researchers from the Common Crawl March 2017 web archive. The evaluation has shown, that this system extracts about 20% of the available scientists with a precision of 80% on real world examples. The implementation is mostly independent of classifiers and raw data origin so that the developed system can be used to evaluate other approaches or data sources in future work.

5.1. Improvement suggestions.

Improving the system should firstly focus on the recall since for this metric the largest improvements are possible. Quick and easy to implement improvements for the classification can be implemented in preprocessing on the text contents.

- **Generous Boilerplate removal:** A main reason for low recall is the lack of features at all as described in Section 4.3.1 for Pages with little content. This could be improved by changing the *JusText* algorithm to be more generous and may worth to be evaluated.
- **Word-Stemming:**[1] As seen in Section 4.2 the heading "Research Interests" is a strong feature in favor of personal pages but it can be assumed that "Research interest" is eaqually important. Word stemming would reduce the amount of distinct features with the same meaning.
- **Structural features**: Features like the number of images, tables, headings might yield additional evidence for classification. Kai Shih et al. [10] also used the table layout of a page to categorize its content which could also be applicable in our task.

More complex improvements might consider using the images on the page for semantic segmentation and the resulting labels as features for page classification. Since

[1] Stemming `https://en.wikipedia.org/wiki/Stemming` [Online; Last access 11.03.2018]

recurrent neural networks are successfully applied in a lot of NLP tasks like Sentiment Classification (Hirschberg et al. [11]) it stands to reason that these techniques can also be applied to the task of webpage classification.

Another starting point is the fact, that webpages exist within a hyperlinked context, with links to and from other pages. An incoming link from a persons index page is a strong feature in favor of a personal page. Hence the labels of webpages linking to or linked from can be used as additional features.

To improve the real world recall we suggest to union the results sets for crawled webpages in Common Crawl which is expected to yield more results as shown in Section 4.1.

6. Acknowledgments

I wish to express my sincere thanks to my supervisor Prof. Dr. Hannah Bast for continuously supporting me in our meetings by providing me with guidance to accomplish the tasks I faced in this work. The willingness to set up meetings at any time and the instant support on all kind of issues where really great!

List of Figures

List of Tables

List of Algorithms

A. Additional Sources

A.1. Gender name Mapping

To build the *name* ⇒ *gender* mapping used in section 3.3.2 we used the following sources to create the mapping:

- https://github.com/ropensci/genderdata
- https://catalog.data.gov/dataset/baby-names-from-social-security-card-applications-data-by-state-and-district-of-
- https://catalog.data.gov/dataset/baby-names-from-social-security-card-applications-national-level-data
- http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/nlp/corpora/names/0.html
- https://usa.ipums.org/usa/
- https://www.nappdata.org/napp/

Combining the sources yielded 97,220 names mapped to their occurences by gender. The mapping and an implementation of our used Algorithm 3 can be found on GitHub.[1]

A.2. Raw evaluation measurements

(TODO: Where are the results stored.)

- Coverage eavaluation results can be found here: `http://www.todo.de`

[1] name-gender-mapping `https://github.com/samuorous/namegender` [Online; last access 27.02.2018]

Bibliography

[1] H. Bast, F. Bäurle, B. Buchhold, and E. Haussmann, "Broccoli: Semantic full-text search at your fingertips," *arXiv preprint arXiv:1207.2615*, 2012.

[2] X. Qi and B. D. Davison, "Web page classification: Features and algorithms," *ACM computing surveys (CSUR)*, vol. 41, no. 2, p. 12, 2009.

[3] M.-Y. Kan and H. O. N. Thi, "Fast webpage classification using url features," in *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM '05, (New York, NY, USA), pp. 325–326, ACM, 2005.

[4] S. D. Gollapalli, C. Caragea, P. Mitra, and C. L. Giles, "Researcher homepage classification using unlabeled data," in *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, (New York, NY, USA), pp. 471–482, ACM, 2013.

[5] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," *Machine learning: ECML-98*, pp. 137–142, 1998.

[6] E. Gutiérrez, "List of all universities in the world in csv," 2015. Available at `https://github.com/endSly/world-universities-csv`.

[7] J. Pomikálek, *Removing boilerplate and duplicate content from web corpora*. PhD thesis, Masaryk university, Faculty of informatics, Brno, Czech Republic, 2011.

[8] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell, "Learning to classify text from labeled and unlabeled documents," in *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pp. 792–799, American Association for Artificial Intelligence, 1998.

[9] J. R. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by gibbs sampling," in *Proceedings of the*

43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005), pp. 363–370, 2005. Available at `http://nlp.stanford.edu/~manning/papers/gibbscrf3.pdf`.

[10] L. K. Shih and D. R. Karger, "Using urls and table layout for web classification tasks," in *Proceedings of the 13th international conference on World Wide Web*, pp. 193–202, ACM, 2004.

[11] J. Hirschberg and C. D. Manning, "Advances in natural language processing," *Science*, vol. 349, no. 6245, pp. 261–266, 2015.