Using Webscraping and Textmining for Constructing the Market Landscape of Firms

Sebastian Heinrich[*], Michael König[†], Timo Schäfer[‡], Martin Wörter[§]

May 10, 2019

1 Data crawling

For each company's domain, a web crawler downloads all textual content and links by considering the content on the landing page and one page behind. For now, we restrict neither the download size of a webpage's content nor the number of webpages that are downloaded until a depth of two is reached. We do not apply any heuristics at this stage to select which content is downloaded, e.g. URL length, such that we ensure that all potentially relevant content is downloaded.

It would be useful to use a proxy in future, such that it is guaranteed that the crawler downloads the content from the English version of the respective domain. Due to the usage of a German IP address, the landing page of large companies is often in German. The crawler handles errors, especially when there is a time-out error or when the crawler is not allowed to connect to a domain. In these cases, the respective company is disregarded from the analysis that we outline in the following sections. For more details on the current implementation and future to-dos, see appendix (A).

Yet, one has to investigate carefully why some webpages experience specific errors - both with and without error messages.

2 Data preprocessing

2.1 Data aggregation and indexing

For each webpage, the crawler creates a csv file that contains the textual content and the URL path of the respective domain. We index each domain and its subpages such that the textual content of each webpage can be easily aggregated and the domain can be more easily cleaned, see section (2.3).

[*]heinrich@kof.ethz.ch
[†]koenig@kof.ethz.ch
[‡]t.schaefer@wiwi.uni-frankfurt.de
[§]woerter@kof.ethz.ch

2.2 General preprocessing

For each webpage, an algorithm classifies the language of every textual data point [2, 1].[1] Then, conditional on that a text's language is English, text is tokenized, uncapitalized, stopwords[2] and punctuation is removed. We decide in favor of term taggers using the fastText[3] library since they are trained on crawled web data that is highly similar to our data on firms' webpages. Afterwards, words are mapped to the respective IDs of the fastText English word embeddings[4] that are used in the subsequent analysis [4].

2.3 Webpage cleaning within and between webpages

For each webpage, there exist several subpages that (potentially) contain irrelevant information without capturing intrinsic, discriminatory details. On the one hand, we want to remove textual content that is too general, e.g. 'All rights reserved', appears on every page (or threshold), and we can therefore identify using simple heuristics. On the other hand, we want to remove the complete textual content from a webpage that contains non-discriminatory information, e.g. login or contact pages. The algorithm we use for this classification problem is described in the appendix (B).

3 Learning firms' product and technology portfolios

3.1 Products

We use companies' NAICS codes[5] (and/or Factset product information) as target variable to learn a firms' product portfolios from the firms' websites. We have primary and secondary NAICS codes for firms, while we formulate a multi-class classification problem for primary NAICS codes and a multi-label classification problem for secondary NAICS codes. For this purpose, we use a Deep Convolutional Neural Network (CNN) as outlined in Figure (1) that comes from [3].

3.2 Technologies

We use patent data, e.g. patent classification IDs and patents' titles, as input for a supervised learning algorithm of firms' technologies. The formulation of the classification problem is analogous to the one for products.

[1] We use following pre-trained model for language identification: https://fasttext.cc/docs/en/language-identification.html

[2] A list of English stopwords comes from https://www.nltk.org/book/ch02.html

[3] https://fasttext.cc/

[4] Words that cannot be found in the fastText vocabulary are removed, but stored in a separate file for further checks, see https://fasttext.cc/docs/en/english-vectors.html.

[5] https://www.naics.com/search/

References

[1] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

[2] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[3] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[4] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

A Data crawling

A.1 Changes (clarifications)

- batch version corrected
- parallelisation implemented (previous version didn't work)
- depth (`DEPTH_LIMIT`): 2 (excluding non-domain links via `LxmlLinkExtractor`), get links and content from main page and one page behind.[6] An alternative specification using number of 'fragments/' is implemented as well, but does not 'yield better results'.
- `PAGE_COUNT_LIMIT`: 1000
- `DEPTH_PRIORITY`: breadth (unchanged, instead of depth)
- `DOWNLOAD_MAXSIZE` set to 1GB (before 5MB)
- add bvdid as variable in text output (mapping from bvdid to webpage isn't bijective)
- obtain *English* version of webpage; if not easily available, throw error and disregard webpage (see to-dos)
- error handling for timeout (might increase waiting time), http and dnslookup error: write error into text.csv file (and into log file)
- html encoding errors corrected (set to "ignored")
- allowed_domains more flexible: include both Orbis url and first request url, see e.g. `cardinalhealth.com` though in Orbis `cardinal.com`
- set `dont_filter` to True (and manually incorporate depth because it is not considered somehow...), otherwise strange behavior because it does not follow all links for every domain
- TimeOutError, when too many requests started
- too many open files error: `ulimit -Sn 40000`

A.2 To-dos

- javascript error, see e.g. *www.jea.com*
- preferred **language** always set to English (sometimes not working via `Accept-Language`, see `microsoft.com`) ⇒ using heuristics and exception handling to circumvent this does not work reliably ⇒ use proxies[7]

[6] Subpages have to *contain* main domain name, e.g. for domain `gm.com`, `media.gm.com` and `gm.com/our-brands` are included. But: subpages `xbox.com` or `channel9.msdn.com` on `microsoft.com` are not "visited".

[7] For instance, https://www.hidemyass.com/de-de/pricing

- what is the **'optimal depth'**? do some manual checks for companies of different size and technology vs. products
- check whether downloaded content from webpages is complete (all links downloaded?) $\Rightarrow$ error handling in case of httperrors, while other parts of domain are downloaded
- potentially only store information from "relevant" tags and identify subpages with granular information by using url information, e.g. #fragments in url (see ARGUS)
- 403 error (access denied), see https://docs.scrapy.org/en/latest/topics/practices.html#avoiding-getting-banned
- change configuration (?): `robots.txt` (`ROBOTSTXT_OBEY` currently set to true)
- (gain better understanding of middlewares)
- (appearance of relative paths?)

B Classification relevant vs. irrelevant webpage content

For the purpose to identify whether a webpage contains relevant or irrelevant information, we construct a data sample using simple heuristics. First, a webpage is labeled as 'relevant' (r) if it contains words such as 'about', 'product' or 'service' in the webpage's URL. Second, a webpage is labeled as 'irrelevant' ($\neg r$) if it contains words such as 'privacy', 'contact' or 'disclaimer' in the webpage's URL.[8] Then, we represent a webpage's textual content in an embedding space to capture the semantic structure of a webpage. This allows us to classify a webpage as r or $\neg r$ conditional on its word embeddings.[9] For an overview of the network architecture from [3], see Figure (1).

[8] The current list of keywords in URLs is as follows: "termsofuse|privacy|data|contact|impressum|search|disclaimer|cookie|investors|site-terms|faqs|compliance|login|support"

[9] For a given webpage, we take the mean of all words' embedding vectors.

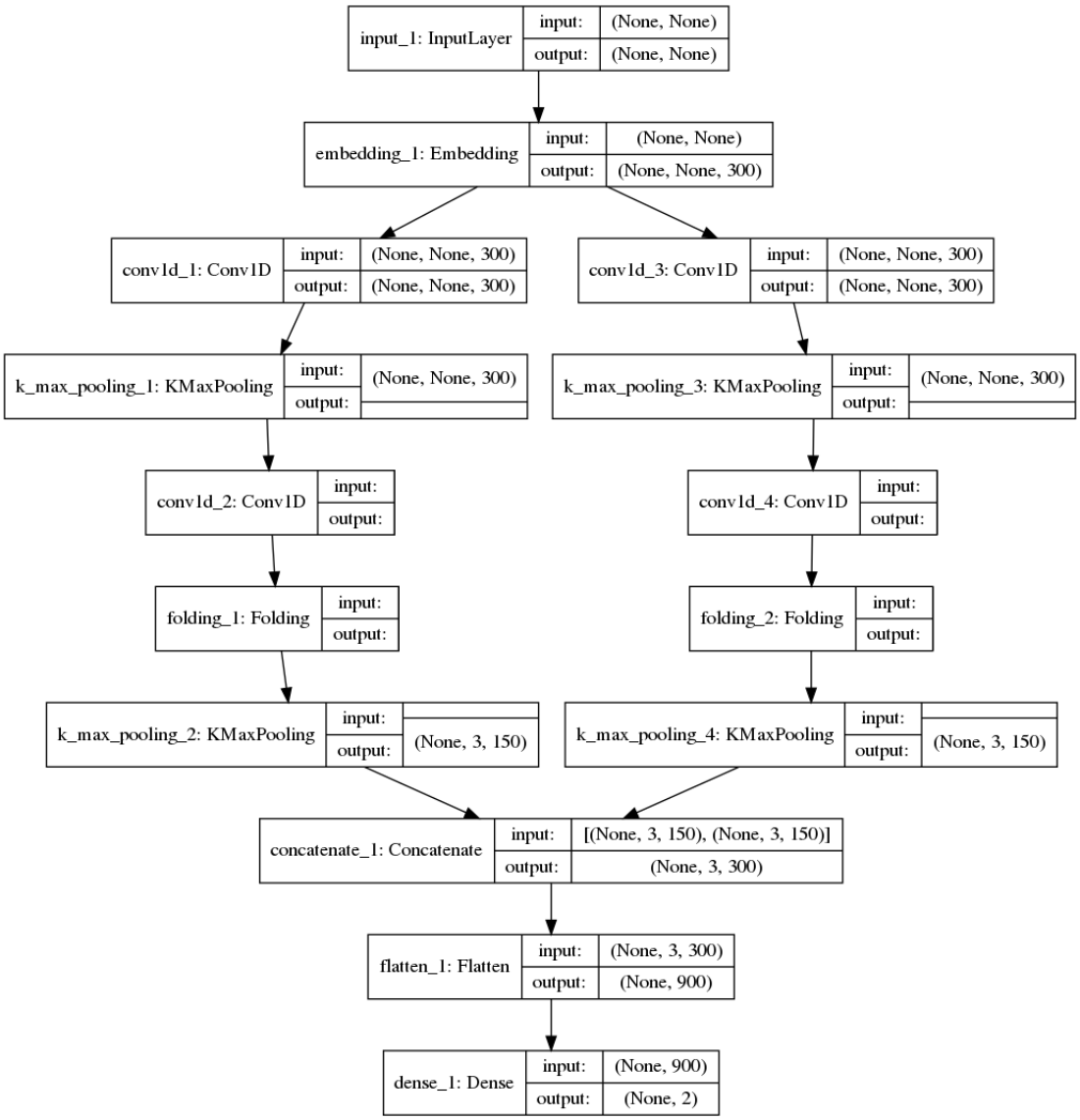

Figure 1: Deep Convolutional Neural Network (CNN) for classification.