News Article Extraction Using Graph Embeddings

Philip Hausner[1], Michael Gertz[1]

[1]Institute of Computer Science, Heidelberg University, Germany

Abstract

Content extraction from web pages is a challenging task due to the heterogeneous nature of the web. In this work, a novel method for the extraction of news articles from arbitrary news article pages is presented that aims to identify the main article content, and removes other elements such as advertisements, navigation elements or comments, that are also commonly present on news article pages. To achieve this, the method utilizes the structure of the DOM tree, which underlies each web page as a hierarchical graph structure, and applies graph representation learning to compute suitable graph embeddings. These graph embeddings are then used to classify web page elements as content or no content, and an additional refinement step then extracts the main article text and removes remaining noise. In the final evaluation on a hand annotated data set collected from 16 German news outlets, we showcase that our method beats all baselines by a significant margin, while only being trained on a comparatively small data set.

Keywords

Graph Representation Learning, Web Content Extraction, Boilerplate Removal

1. Introduction

The content that is published on the World Wide Web increases with every day, and many downstream applications can benefit from utilizing this content in an efficient way. However, web pages most often contain more than just the main content a user is interested in. Consider a typical news article page: Usually, not only the article text itself is displayed, but the web page additionally incorporates functional elements like menu bars, advertisements, or published user comments. This circumstance demands for methods that are able to extract content efficiently, and to filter elements that do not represent meaningful content or that are irrelevant for respective downstream tasks. However, due to the high heterogeneity of web pages and their complex structure, content extraction from web pages remains to be a challenging task. The most prominent common point in the architecture of web pages is their structure as a DOM tree. The DOM tree is a hierarchical tree structure in which each element of the web page is represented as a node in the tree. In this work, the structure of a web page as a graph is utilized to extract content from a web page. Graphs are among the most prevalent and versatile data structures, and can be a useful representation for many use cases besides web pages, e.g., knowledge graphs or social networks. As a result, recent research has increasingly investigated means to employ machine learning on graph structures. There exists a wide range of graph analysis tasks [1, 2, 3, 4, 5], but among the most common tasks are link prediction, e.g., to predict

LWDA 2021: Lernen. Wissen. Daten. Analysen. - Learning. Knowledge. Data. Analytics., September 01–03, 2021, Munich, Germany

hausner@informatik.uni-heidelberg.de (P. Hausner); gertz@informatik.uni-heidelberg.de (M. Gertz)

connections of entities in a knowledge graph [6], and node classification to identify nodes with similar structural roles [2]. Leveraging graph machine learning proves to be a promising approach for many application fields: Gaudelet et al. [7] recently investigated in which way drug discovery can benefit from graph machine learning, Wu et al. [8] utilized graph convolutional networks to detect social spammers, and Fout et al. [9] used graph convolutional networks to predict protein interfaces. A basis for these downstream tasks are often low dimensional vector embeddings, from now on called graph embeddings, for each node in a potentially large graph that combine knowledge about the node itself with knowledge about the node's neighbourhood, hence, utilizing the topology of the graph. In this work, we discuss the usage of graph embeddings in the context of content extraction. In particular, we focus on the extraction of news article texts from a range of German news outlets. Therefore, we present an approach to utilize graph embeddings in the context of web pages by taking advantage of the structure of a HTML page in the form of the DOM tree. We first build suitable vector representations for each node of a web page using graph convolutional networks, and then outline how these vectors can be employed for further downstream tasks such as content extraction.

Our contributions include:

- We present a graph convolutional network architecture that embeds web elements in a low dimensional vector space.
- We introduce a new algorithm to extract content elements from a web page by utilizing the aforementioned node embeddings.

The paper is structured as follows: First, related work is presented that both discusses web content extraction as well as graph representation learning. Second, the extraction of features from web pages is discussed. Third, the graph convolutional model this work evolves around is presented, and an additional refinement is introduced that is suited for the extraction of article text from web pages. Lastly, results are presented and a short conclusion is given.

2. Related Work

Web Content Extraction. Web content extraction is the task of identifying and extracting the main content of a web page and distinguish it from the boilerplate, a term that collectively describes page elements such as navigation bars or advertisements. Most content extraction and boilerplate removal algorithms leverage features of the DOM tree in combination with heuristics or machine learning methods in order to identify boilerplate elements. Kohlschütter et al. [10] introduced *Boilerpipe*, a popular method to identify boilerplate using shallow text features, which is still widely used, and was integrated into the web crawler Apache Nutch[1]. Another heuristic-based approach for boilerplate removal was introduced by Pomikálek [11, p. 29] and is known as *jusText*. Wu et al. [12] formulated the main content detection problem as a DOM tree node selection task, and utilized DOM tree node properties as input for a machine learning model. Additionally, they implemented a grouping mechanism to identify the main content and distinguish it from noisy fragments. A more recent approach by Vogels et al. [13] leverages a hidden Markov model and a neural network architecture to extract main content from a web

[1] http://nutch.apache.org/

page. Sirsat and Chavan [14] employed a pattern matching technique to extract contents from news web pages using mainly regular expressions. While the approaches described in this context mostly aim at removing boilerplate, we formulate a stricter task in this work, and aim to extract the main article text from German news articles, and hence, want to remove content, e.g., user comments, that is generally not considered to be boilerplate as well.

Graph Representation Learning (GRL). Due to its success in various application domains, graph representation learning has become a research field of high interest. One can separate methods for GRL into various groups, most prominently supervised and unsupervised methods. Unsupervised techniques aim primarily to capture the graph structure, and hence, can, for example, identify nodes with similar structural roles. Probably the most prominent unsupervised methods are Deepwalk [3] and node2vec [2], which adapt the word2vec model introduced by Mikolov et al. [15] to be applicable for graph-structured data. This is achieved by performing fixed-length random walks on the graph, and treating the sequence of visited nodes as a sentence input for the word2vec model. Deepwalk and node2vec mainly differ only in the fact that while Deepwalk utilizes unbiased random walks, node2vec introduces an additional hyperparameter that influences the employed graph exploration strategy. Specifically, node2vec adapts the strategy by either guiding the underlying random walks to focus on the local neighbourhood of a starting node (similar to breadth first search), or to focus on nodes that have a greater distance to the starting node (similar to depth first search).
Regarding (semi-)supervised methods for GRL, a wide range of models was developed during the recent years. In 2016, Kipf and Welling [16] introduced a scalable variant of graph convolutional networks (GCN), a neural network architecture that works on graph representations, that significantly advanced the state-of-the-art at the time. Hamilton et al. [17] proposed GraphSAGE, which instead of taking into account all neighbours of a node, only samples a fixed number of neighbours to compute node embeddings, and hence, reduces computational complexity of previous models. Velickovic et al. [18] added a self-attention mechanism to graph convolutional networks, and coined the term graph attention networks (GAT). Besides these highly influential works, there is a plethora of work investigating possibilities to compute suitable graph embeddings for a wide variety of tasks, and we refer the interested reader to the thorough surveys by Chami et al. [19] and Zhou et al. [20].

3. Feature Selection from DOM Tree Nodes

In this section we elaborate on the initial feature selection step in which for each element of a web page, a feature vector is initialized. Since each element can be represented by at least one node in the DOM tree, see Figure 1, this structure can be directly utilized, and we assign a feature vector to every node of the DOM tree. These vectors are the input for the subsequent processing using a graph convolutional network to calculate lower dimensional embedding vectors. We consider the following properties of each DOM tree node.

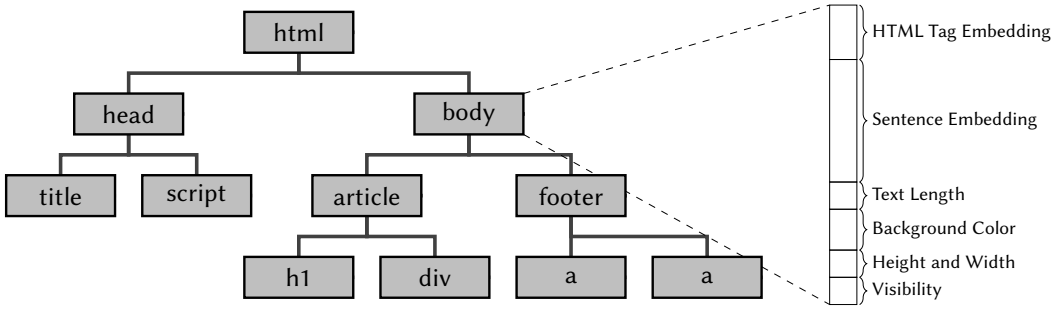

Figure 1: Example of a DOM tree (left), and the corresponding feature vector representation associated with the body element (right).

HTML Tag. Each node in the DOM tree has an associated HTML tag that can be encoded in a vector representation. While one could simply use a one-hot encoding for each distinct HTML tag, we precompute suitable HTML tag embeddings beforehand. To achieve this, we compute the set of all pairs of nodes from our training data that are directly connected by an edge in the DOM tree, and then extract the respective HTML tags from the nodes. This means, given two nodes n_1 and n_2 directly connected by an edge, the tuple $(n_1.html\_tag, n_2.html\_tag)$ is part of the computed set. Similar to Deepwalk [3] these tuples can now be interpreted as sentences and are used as input for a word2vec model [15], hence, yielding a distinct vector representation for each HTML tag. However, note that while Deepwalk creates random walks using a fixed size, our approach does not randomly sample any sentences, but creates them deterministically using the DOM structure, and only captures the direct neighbourhood of each node. Due to the comparatively low number of existing HTML tags, the dimension of HTML tag embedding vectors is restricted to 8 in the experiments outlined in Section 6.

Sentence Embeddings. Sentence embeddings for each text node are computed by employing a simple and fast sentence embeddings method introduced by Arora et al. [21] and implemented by Borchers [22]. As initial training data for the underlying word2vec model, five million sentences from German news (from 2013 to 2015 as well as from 2018 and 2019) and one million sentences from Wikipedia (2016) were chosen from the publicly available *Leipzig Corpora Collection* [23]. The size of vector embeddings is set to 200, and for nodes containing no text, the null vector is set manually.

Text Length Indicator. The length of the text contained by a DOM tree node. Since only text nodes are considered for textual content, this value is 0 in most cases. To reduce the large difference in values, this indicator is scaled by 0.01, which led to slightly better results in the following experiments.

Color. The color feature consists of the three values of the element's RGB background color value scaled to values between 0 and 1.

Relative Height and Width. The height and width of each element are, respectively, divided by the total height and width of the given web page, and hence, the relative height and width is added to each element's feature vector. As a result, this value is a number between 0 and 1, representing the fraction of the element's height/width with regard to the height/width of the whole web page.

Visibility. This is a Boolean value that is set to 1 if the element's *display* property is not *"none"* and the height and width are larger than 0. Otherwise the value is set to 0.

The resulting feature vector for each DOM tree node is the concatenation of all above values, resulting in a vector of dimension 215 as indicated in Figure 1.

4. Content Classification Network

In this section, we present a graph convolutional network (GCN) architecture to compute suitable vector representations for downstream classification tasks on web pages. The goal is to build denser low-dimensional vector representations that do not only incorporate information about the node itself, but also about its neighbourhood. Therefore, the structure of the DOM tree as a hierarchical tree structure is utilized. The DOM tree is a graph representation in which each node, except for the root, has exactly one parent node, and an arbitrary number of children. As a result, methods from graph representation learning as presented in Section 2 can be applied to the DOM tree as well. In this work, an adapted version of the architecture by Kipf and Welling [16] is utilized. Kipf and Welling propose a GCN with the following layer-wise propagation rule:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{1}$$

with $H^{(l)}$ being the matrix of activations at layer l, and $H^{(0)} = X$ being the initial matrix of feature vectors. A is the adjacency matrix of a given undirected graph, and $\tilde{A} = A + I$ is the adjacency matrix with I being the identity matrix adding self-connections, $\tilde{D}_{ii} = \Sigma_j \tilde{A}_{ij}$ a normalization matrix, $W^{(l)}$ the trainable weight matrix for layer l, and σ a nonlinear activation function such as ReLU.

In this work, this architecture is modified as follows. Firstly, in the proposed model A is the adjacency matrix of a directed graph in which A_{ij} equals 1 if node j is the child of node i in the DOM tree, assuming a numbering of nodes from 1 to n with n being the number of DOM tree nodes on a given web page. This change is necessary, since the DOM tree is a directed acyclic graph that implies hierarchical relations between elements, and hence, parent nodes should generally be treated differently from child nodes. We then define $Z^{(l)}$ at layer l as:

$$Z^{(l)} = \texttt{CONCAT}\,(H^{(l)}, \hat{A}H^{(l)}, \hat{A}^T H^{(l)}) \tag{2}$$

with $\hat{A} = \tilde{D}^{-\frac{1}{2}} A \tilde{D}^{-\frac{1}{2}}$, and $\tilde{D}$, $H^{(l)}$ and I defined as above. It again holds that $H^{(0)} = X$, the initial feature vectors. Note that in this case A has no added self-loops. Intuitively, $H^{(l)}$ is just the matrix of activations at each node at layer l, $\hat{A}H^{(l)}$ is the aggregation of all child nodes for each DOM tree node, and $\hat{A}^T H^{(l)}$ is the activation vector of the parent node (note that in a DOM tree each node has at most one parent node). For each node, these three vectors are then concatenated. Finally, the layer-wise propagation rule is altered to:

$$H^{(l+1)} = \sigma(Z^{(l)} W^{(l)}), \tag{3}$$

and in this work more specifically to

$$H^{(l+1)} = ReLU(ReLU(Z^{(l)} W_0^{(l)}) W_1^{(l)}). \tag{4}$$

Given the number of layers l (often also called *iterations*) set to $l = m$, this yields $H^{(m)}$, the final embedding vectors. For the final classification task, an additional layer is introduced that maps each embedding vector to a class label:

$$Y = \sigma(H^{(m)} W_c). \tag{5}$$

Since this work focuses on the identification of main content on a web page, this is a binary classification task, and each entry at index i of Y is a single value indicating whether node i is content or not. To achieve this, σ is a sigmoid function, and for final classification, a suitable threshold value is chosen that distinguishes between positive and negative samples. During training, the binary cross entropy loss function is minimized over all labeled training examples.

5. Refinements of Initial Results

Inspection of news article pages quickly leads to the insight that articles are usually blocks of text that are in close proximity to each other, only sometimes separated by interjecting blocks, such as images or advertisements. Additionally, one never finds pieces of article content scattered throughout a single page. This insight can be utilized to refine the results yielded by the initial classification as given by Equation (5). This refinement is implemented by an adaption of the grouping algorithm introduced by Wu et al. [12], which groups candidate elements into different groups by utilizing spatial information about the elements. After this separation into multiple groups, Wu et al. then implement an additional method to select the group that most likely encompasses the main content. However, in this work, a simplified version is presented that is better suited for the domain at hand, namely news article pages. The method is subdivided into two steps, first grouping and then selection, in which in the first step the initial results are merged into a set of groups where each node belongs to one group, and in the second step the algorithm selects the group that is most probably the main article content of the page.

First, the candidate elements are sorted from top to bottom with regard to their appearance on the vertical axis, such that for a sequence N containing all candidates, it holds that node n_i is above or on the same level as node n_j on the web page if $i < j$ and $n_i, n_j \in N$. If both nodes n_i and n_j are on the same level of the vertical axis, they are sorted in arbitrary order. In the following, we call the coordinate that specifies the element with respect to its height on the web page the y-coordinate of the element. The sequence N is then separated into multiple groups by defining breakpoints between nodes. Given the y-coordinate y_i and height h_i of a node n_i, and a predefined threshold t, a breakpoint is inserted between two consecutive nodes n_i and n_{i+1} if the following condition does not hold:

$$y_{i+1} - (y_i + h_i) \leq t. \tag{6}$$

Intuitively, this means that the distance between the lower edge of the upper element n_i and upper edge of the lower element n_{i+1} cannot be larger than the threshold t if n_i and n_{i+1} belong to the same group. This is in accordance with the intuition that in a news article, the main article text is usually one or multiple text blocks near to each other, while elements such as user comments are separated from the main article.
Finally, given the set of groups G, we select the group $g \in G$ for which the text length is maximal. This assumes that the longest block of text identified by the applied grouping is also the article itself, while other text blocks are (much) shorter, which holds true in most cases. Potential exceptions to this are cases in which the content classifier detected large parts of the comment section as main content. However, experiments showed that this is only rarely the case, and proofed to be only problematic for very short news articles.

6. Experimental Evaluation

In the following, the presented model is evaluated on a labeled data set. First, the data set is introduced, second, implementational details are discussed, third, baselines are introduced, and last, results are presented. Note that for reported models the best result over 5 runs is reported.

Evaluation Data Set. While there are existing data sets to evaluate boilerplate removal and content extraction, such as CleanEval [24] (741 English and 713 Chinese documents) and L3S-GN1[2] (621 English documents), they have considerable disadvantages. Firstly, they are outdated: CleanEval was published in 2008 and L3S-GN1 in 2010, and since then the structure of web pages has changed significantly. Secondly, they lack meta descriptors, like the color or size of DOM elements, which are essential for the model proposed in this work. And thirdly, both data sets do not provide resources for the German language which is targeted in this work. To the best of our knowledge, no other data sets, that are suitable to evaluate the task at hand, are freely available, and therefore, we annotated our own data set. For training 131 web pages from 16 German news outlets were annotated during May 2021 by the first author of this work. Tests were conducted on a second set also annotated during May 2021 by the first author, consisting of 73 articles from the same 16 news outlets. For every training point, a complete

[2] http://www.l3s.de/%7Ekohlschuetter/boilerplate/, accessed July 16, 2021

page containing a news article was manually annotated, labeling every block of text that clearly was part of the main article text or a heading while elements like *info boxes*, advertisements or images were ignored. If a block of text was wrapped by multiple HTML elements that were equally suitable to be labeled as positive, only one was annotated. This takes into account that annotations are often conducted by laymen that have no background knowledge about the methods that are later on applied to the data, and hence, intuitively only annotate elements they consider to be content once. Therefore, this data set also tests if a given method performs well for noisy annotations where not each positive sample is labeled as such. For training and test setting, every element that was not labeled positive was treated as a negative sample. While this is not entirely accurate due to the annotation process, it can be assumed that the influence of samples falsely labeled as negative is limited, especially since positive labels were weighted higher during training. This was particularly necessary, because out of 193,635 elements on the web pages, only 1737 had a positive label. A complete list of the chosen German news outlets, and an example annotation can be found in Appendix A.1 and Appendix A.2.

Implementation Details. The initial training set was split further into a set of 111 web pages used for training, and a development set consisting of the remaining 20 pages. The GCN was initialized, such that the dimension of embeddings (except for the initial size of the feature vectors) was of size 20 and the ADAM optimizer with Pytorch's default parameters and learning rate of 10^{-4} was used for backpropagation. The model was trained over a maximum of 40 epochs, however, if the calculated loss increased in three consecutive epochs on the development set, the training was stopped early. During training a dropout of 0.3 is applied in each layer. To address the high inequality in the amount of positive and negative samples, negative samples were weighted by 0.01. Furthermore, since content extraction in this context was formulated as a binary classification problem, but Equation (5) yields a value between 0 and 1, a suitable threshold has to be defined. In this case, all values in 0.025 steps between 0 and 1 were evaluated on the F1 score of the development set, and the value maximizing this metric was chosen as threshold for the model. The threshold t for the refinement step as introduced in Equation (6) is set to 1000 pixels. Finally, in this work one model was trained using two layers, and one using three layers, i.e., m as specified in Section 4 was either 2 or 3. Intuitively, this means that for a node n all nodes have an influence on the final embedding vector of n that have at most a distance of 2 or 3 to n, respectively. We denote the model as either *GCN2* or *GCN3*.

Metrics. To assess the quality of results, first precision, recall and F1 score are evaluated. We report both macro as well as micro averages to address the high inequality in the number of positive and negative samples. Additionally, the minimum edit distance between the ground truth text given by the annotations and the text extracted by the algorithm is determined for each page, and the average is also reported for each model. The minimum edit distance computes the minimum number of editing operations (insertions, deletions and substitutions) that are needed to transform one string into another. In this work, Levenshtein distance is applied, i.e., a substitution accounts for two edit operations. This takes into account that while the problem is framed as a binary classification problem in this work, the goal is still to retrieve as much text as possible from the web page.

	Precision		Recall		F1	
	Micro	Macro	Micro	Macro	Micro	Macro
SVC	1.0	**0.90**	1.0	0.82	1.0	0.86
MLP	0.99	0.79	0.99	0.83	0.99	0.81
GCN2	1.0	0.89	1.0	0.95	1.0	**0.92**
GCN3	1.0	0.87	1.0	**0.97**	1.0	**0.92**

(a)

	Ø Min. Edit Distance
SVC	80.6
MLP	92.0
GCN2	**51.1**
GCN3	53.0
jusText	325.7
Boilerpipe	233.6

(b)

Table 1
(a) Precision, recall and F1 for every model in the form (micro average / macro average). (b) Average minimum edit distance between ground truth text and text predicted by respective model over all test web pages (lower is better).

Baselines. For classification two baselines are implemented. First, a support vector classifier (SVC) using a radial basis function is employed. The second baseline is a multi layer neural network (MLP) using three layers, a total of 128 nodes per layer, and otherwise the same parameters as specified for the GCN models. The result of both baseline classification models is also processed by the refinement method described in Section 5. Furthermore, two established methods for boilerplate extraction are compared to the proposed model. First, Boilerpipe by Kohlschütter et al. [10], and second jusText by Pomikálek [11]. For both methods, readily available Python implementations exist[3]. Note that for Boilerpipe, we use the *ArticleSentencesExtractor*, which is specifically adapted for news articles. Due to the different architecture of both Boilerpipe and jusText that does not yield a classification result for each node, only the minimum edit distance is reported.

Results. Table 1a shows the classification results for all four models. One can see that while the precision of both GCN models does not improve over the performance of both baselines, recall increases significantly, and as a result the F1 score is better as well. However, as we addressed before often multiple DOM tree nodes can be a valid result if the goal is to retrieve the main article text of a web page, and hence, the F1 score can only indicate the quality of results. Table 1b completes the picture by giving insights into the average minimum edit distance between the ground truth texts and the predicted texts by each model. One can see that the heuristic methods jusText and Boilerpipe perform significantly worse than all of the models utilizing machine learning. However, one should also note that a fairer comparison between the heuristic and machine learning approaches would be on a domain all models have not seen before, e.g., a news outlet that was previously not part of the data set. Additionally, it can be seen that the graph convolutional methods yield text that is closer to the ground truth as the result of both baselines. This is a strong indication that the context of a node has an influence on the classification as well, and including this into our models can improve classification significantly. To address the question why the F1 score between the GCN and (at least the SVC) baseline models does differ only slightly, but the minimum edit distance reflects a significantly

[3] https://pypi.org/project/boilerpy3/ and https://pypi.org/project/jusText/, accessed July 16, 2021

better result in case of the GCN models, a manual inspection was conducted as well. This inspection showed that while the baseline models struggled to differentiate between the main article content and subtle advertisements or info boxes between the article text blocks, the GCN models often detected high quality comments of users that, if given without the position on the page, were hardly distinguishable from text written in a news article. However, since the comment section is most often clearly separated from the main article, the refinement step described in Section 5 was able to remove these text blocks from the final result, hence, yielding better overall results for the graph convolutional models.

7. Conclusion and Ongoing Work

In this work, we have introduced a graph convolutional model to classify DOM tree elements on web pages, and showcased its application in the context of content extraction from German news pages. However, the architecture of the presented model is general enough, such that it can be applied to various classification tasks on web pages, even multilabel and multiclass classification tasks by adaption of Equation (5), as long as sufficient training data is available. In particular, it was shown that even for a skewed data set with only few positive samples this model already yields promising results. This can also be of use in other application areas where the goal is to detect rare patterns on web pages, for example, Dark Patterns that aim to trick users into actions that are against their best interest. Besides the application of the proposed model in other domains, future work can also include the addition of more features in the initial feature selections step, see Section 3. Alternatively, the computation of HTML tag embeddings could be replaced by embeddings computed by Deepwalk or node2vec. However, this is of course highly dependent on the task and the domain at hand. Finally, one problem of the proposed model is that it only works for static graph representations. In cases where dynamic content is more relevant, it may be more suitable to resort to methods like GraphSage [17] that can better include the dynamic addition or removal of nodes from a graph, since they do not utilize the adjacency matrix of the graph for embedding calculation, but instead work with an adjacency list.

Acknowledgments

This work is part of the Dark Pattern Detection Project (dapde), which is funded by the German Federal Ministry of Justice and Consumer Protection. More information on the project can be found under https://dapde.de.

References

[1] S. Cao, W. Lu, Q. Xu, Grarep: Learning graph representations with global structural information, in: Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015, ACM, 2015, pp. 891–900. URL: https://doi.org/10.1145/2806416.2806512. doi:10.1145/2806416.2806512.

[2] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016, ACM, 2016, pp. 855–864. URL: https://doi.org/10.1145/2939672.2939754. doi:10.1145/2939672.2939754.

[3] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: online learning of social representations, in: The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014, ACM, 2014, pp. 701–710. URL: https://doi.org/10.1145/2623330.2623732. doi:10.1145/2623330.2623732.

[4] M. Togninalli, M. E. Ghisu, F. Llinares-López, B. Rieck, K. M. Borgwardt, Wasserstein weisfeiler-lehman graph kernels, in: Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, 2019, pp. 6436–6446. URL: https://proceedings.neurips.cc/paper/2019/hash/73fed7fd472e502d8908794430511f4d-Abstract.html.

[5] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, J. Leskovec, Graph convolutional neural networks for web-scale recommender systems, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018, ACM, 2018, pp. 974–983. URL: https://doi.org/10.1145/3219819.3219890. doi:10.1145/3219819.3219890.

[6] I. Chami, A. Wolf, D. Juan, F. Sala, S. Ravi, C. Ré, Low-dimensional hyperbolic knowledge graph embeddings, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020, Association for Computational Linguistics, 2020, pp. 6901–6914. URL: https://doi.org/10.18653/v1/2020.acl-main.617. doi:10.18653/v1/2020.acl-main.617.

[7] T. Gaudelet, B. Day, A. R. Jamasb, J. Soman, C. Regep, G. Liu, J. B. R. Hayter, R. Vickers, C. Roberts, J. Tang, D. Roblin, T. L. Blundell, M. M. Bronstein, J. P. Taylor-King, Utilising graph machine learning within drug discovery and development, CoRR abs/2012.05716 (2020). URL: https://arxiv.org/abs/2012.05716. arXiv:2012.05716.

[8] Y. Wu, D. Lian, Y. Xu, L. Wu, E. Chen, Graph convolutional networks with markov random field reasoning for social spammer detection, in: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, AAAI Press, 2020, pp. 1054–1061. URL: https://aaai.org/ojs/index.php/AAAI/article/view/5455.

[9] A. Fout, J. Byrd, B. Shariat, A. Ben-Hur, Protein interface prediction using graph convolutional networks, in: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, 2017, pp. 6530–6539. URL: https://proceedings.neurips.cc/paper/2017/hash/f507783927f2ec2737ba40afbd17efb5-Abstract.html.

[10] C. Kohlschütter, P. Fankhauser, W. Nejdl, Boilerplate detection using shallow text features, in: Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010, ACM, 2010, pp. 441–450. URL: https://doi.org/10.1145/1718487.1718542. doi:10.1145/1718487.1718542.

[11] J. Pomikálek, Removing boilerplate and duplicate content from web corpora, Ph.D. thesis, Masarykova univerzita, Fakulta informatiky, 2011.

[12] S. Wu, J. Liu, J. Fan, Automatic web content extraction by combination of learning and grouping, in: Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015, ACM, 2015, pp. 1264–1274. URL: https://doi.org/10.1145/2736277.2741659. doi:10.1145/2736277.2741659.

[13] T. Vogels, O. Ganea, C. Eickhoff, Web2text: Deep structured boilerplate removal, in: Advances in Information Retrieval - 40th European Conference on IR Research, ECIR 2018, Grenoble, France, March 26-29, 2018, Proceedings, volume 10772 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 167–179. URL: https://doi.org/10.1007/978-3-319-76941-7_13. doi:10.1007/978-3-319-76941-7_13.

[14] S. Sirsat, V. Chavan, Pattern matching for extraction of core contents from news web pages, in: 2016 Second International Conference on Web Research (ICWR), IEEE, 2016, pp. 13–18.

[15] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, in: 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings, 2013. URL: http://arxiv.org/abs/1301.3781.

[16] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, CoRR abs/1609.02907 (2016). URL: http://arxiv.org/abs/1609.02907. arXiv:1609.02907.

[17] W. L. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, 2017, pp. 1024–1034. URL: https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html.

[18] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, CoRR abs/1710.10903 (2017). URL: http://arxiv.org/abs/1710.10903. arXiv:1710.10903.

[19] I. Chami, S. Abu-El-Haija, B. Perozzi, C. Ré, K. Murphy, Machine learning on graphs: A model and comprehensive taxonomy, CoRR abs/2005.03675 (2020). URL: https://arxiv.org/abs/2005.03675. arXiv:2005.03675.

[20] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, AI Open 1 (2020) 57–81. URL: https://doi.org/10.1016/j.aiopen.2021.01.001. doi:10.1016/j.aiopen.2021.01.001.

[21] S. Arora, Y. Liang, T. Ma, A simple but tough-to-beat baseline for sentence embeddings, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, OpenReview.net, 2017. URL: https://openreview.net/forum?id=SyK00v5xx.

[22] O. Borchers, Fast sentence embeddings, https://github.com/oborchers/Fast_Sentence_Embeddings, 2019.

[23] D. Goldhahn, T. Eckart, U. Quasthoff, Building large monolingual dictionaries at the leipzig corpora collection: From 100 to 200 languages, in: Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012, European Language Resources Association (ELRA), 2012, pp. 759–765. URL: http://www.lrec-conf.org/proceedings/lrec2012/summaries/327.html.

[24] M. Baroni, F. Chantree, A. Kilgarriff, S. Sharoff, Cleaneval: a competition for cleaning web

pages, in: Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008, 26 May - 1 June 2008, Marrakech, Morocco, European Language Resources Association, 2008. URL: http://www.lrec-conf.org/proceedings/lrec2008/summaries/162.html.

A. Appendix

A.1. List of German news outlets chosen for annotations

augsburger-allgemeine.de, faz.net, focus.de, freiepresse.de, golem.de, handelsblatt.de, heise.de, n-tv.de, rp-online.de, spiegel.de, sueddeutsche.de, t-online.de, tagesschau.de, waz.de, welt.de, zeit.de

A.2. Example annotation

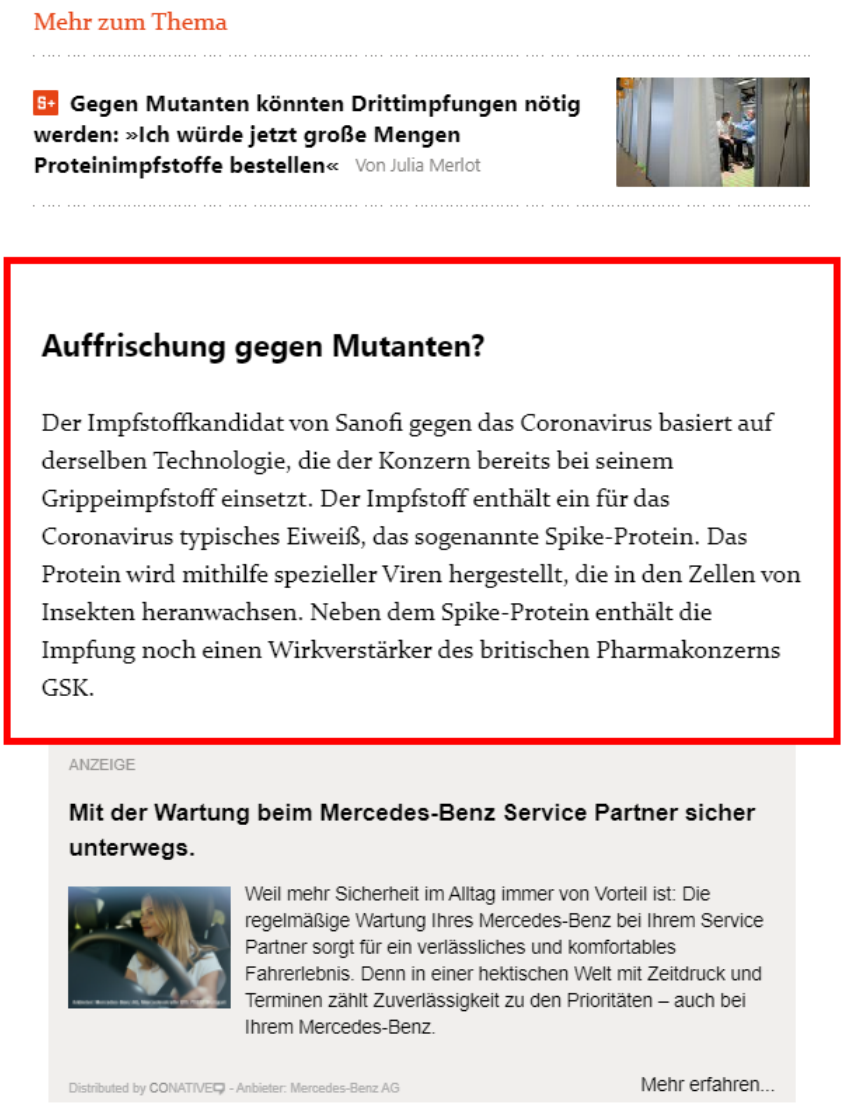

Figure 2: Extract from an annotation from spiegel.de. The red box indicates the part annotated while advertisements and the info box are left out.