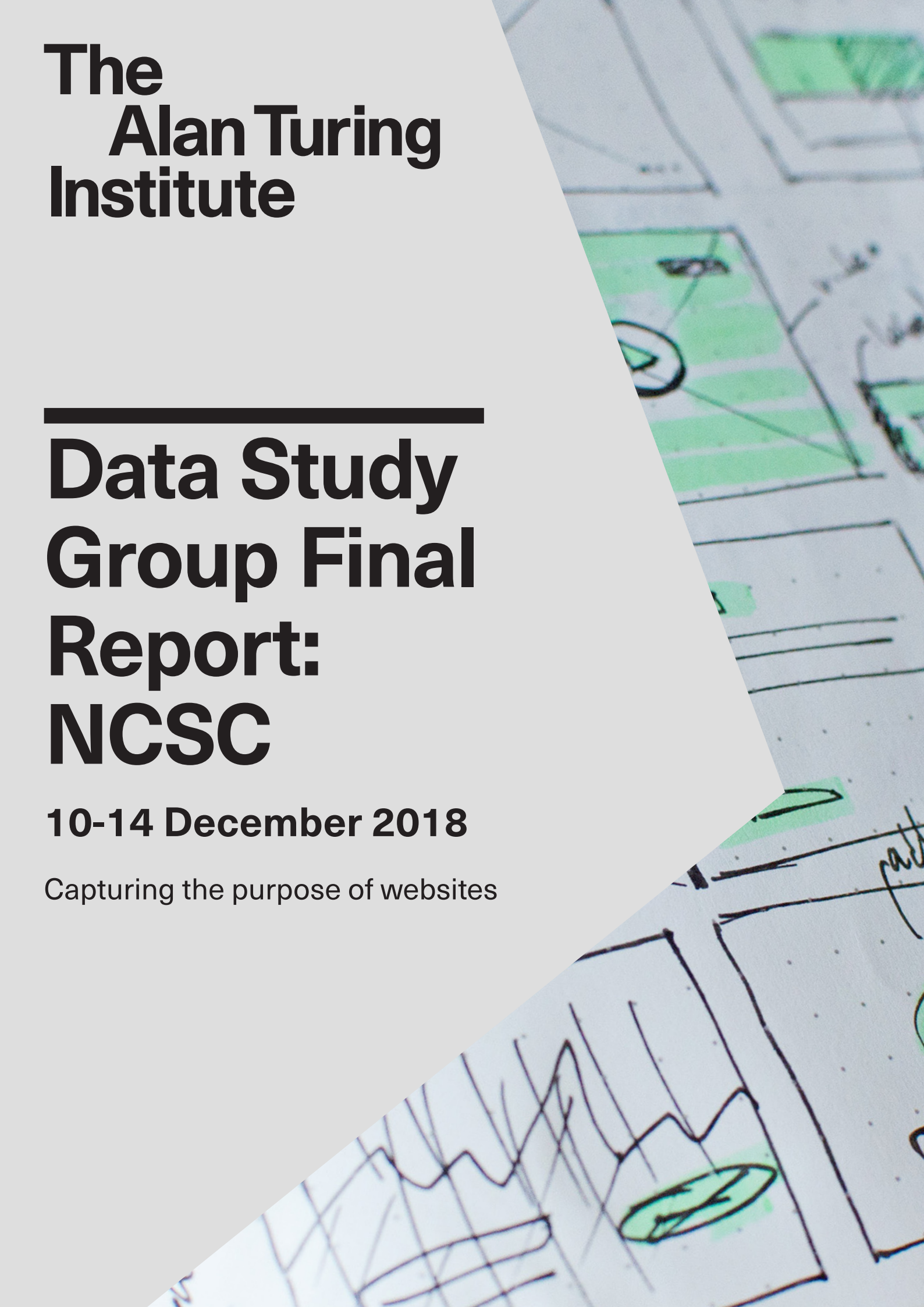
The Alan Turing Institute

Data Study Group Final Report: NCSC

10-14 December 2018

Capturing the purpose of websites



https://doi.org/10.5281/zenodo.3367414

This work was supported by The Alan Turing Institute under the EPSRC grant EP/N510129/1

Contents

1 Executive Summary

This challenge was provided by the National Cyber Security Centre (NCSC), which is tasked with protecting the UK public sector, and aims at improving existing internet search technologies.

1.1 Challenge Overview

As the internet grows, there is a need in industry, government and academia to better facilitate website recommendation, semantic search, and domain discovery, as well as to improve the security of the web. For instance, it is not possible to easily find all UK public sector domains with a simple search; nor it is possible for commercial organizations to easily generate a list of potential competitors or suppliers/customers from current search engines. An exciting potential approach to enable the NCSC and other organizations to leverage the latest machine learning algorithms on these challenges is to automatically learn a purposeful compact vector representation of every website or domain on the Web.

1.2 Main Objectives

This challenge main goal is to produce highly scalable algorithmic methods to represent internet domains as simple numerical vectors. The aim is to investigate this approach through state-of-the-art text and graph embedding techniques from the fields of machine learning, deep learning, and graph theory. These vectors could then be used as the input for a variety of follow-on machine learning algorithms, to be deployed against problems such as: website "description/label" prediction (to supplement hand-crafted descriptions, such as in Curlie[1]), and missing hyperlink prediction.

[1] Curlie is the 'largest human-edited directory of the Web', and a successor to the Open Directory Project (ODP) and DMOZ: `http://www.curlie.org`

Solving these would enable a variety of applications across industry, academia and government including:

- recommending domains (especially businesses) as potential service providers or customers;
- recommending government or other helpful services to individuals or organizations;
- triaging search results by semantic similarity (rather than keyword matches);
- seed-based exploration of the web.

This work also lays a foundation for future work in better securing the internet, where we would like to build a more detailed and systematic understanding of hosts and domains, and how they connect and interact with each other, with a view to future models that can better identify malicious domains and activity on the Web.

1.3 Overview of the Data

For this challenge, we prepared several datasets, consisting of sampled hyperlink web-graphs and website content, derived from open-source web crawl data produced by the Common Crawl project. These datasets were designed to allow rapid prototyping and evaluation of algorithm variants, and were based on a combination of UK domains, and hand-labelled domains from the Curlie project.

1.4 Approaches

We began by preparing strong baselines using several text-only embedding approaches for summarising website content, including LDA, doc2Vec, FastText, ELMo and BERT, as well as several approaches for creating graph embeddings, including node2vec, and Graph Convolutional Networks (GCNs).

We then combined these approaches in order to train hybrid embeddings, using both text contents and graph structure, using primarily the state-of-the-art method GraphSAGE.

Also, we investigated and implemented a novel method able to make any embedding method scale, adopting a Local-to-Global approach. We implemented the pseudo-code in Python, and carried out a preliminary test using synthetic data. We leave as future work to refactor this implementation and test it on real-world datasets. This method may provide a way of scaling very expensive embedding methods to large graphs. It might even enable representations to be created for the 100M web domains, or even the 1B+ individual website hosts that are part of the full CommonCrawl dataset.

1.5 Main Conclusions

Experimenting with the approaches described in the previous subsection we found that on the smaller datasets both text and graph based approaches perform very well on the label classification problem. Moreover we had preliminary evidence that hybrid embedding methods, in our case GraphSAGE, can outperform both at the same task.

On the biggest datasets it was possible to obtain very good classification results again using text embeddings, but it was not possible to employ graph embeddings due to time constraint. The problem of scalability could potentially be resolved by use of the above mentioned Local-to-Global technique, but this needs further investigation.

The most promising approach at this point is a combination of FastText embeddings with a modified version of GraphSAGE that should be able to scale to very large networks with high performance, given some constraints on the maximum degree of nodes in the graph. Again, this constraint could potentially be solved with the Local-to-Global approach.

1.6 Limitations

The work conducted during the Data Study Group week was promising. However, due to the limited time available to run experiments, it was not possible to obtain a thorough comparison of the proposed approaches on the given datasets. For instance, we could not assess the performance of the Local-to-Global method on real-world data, and so there was not way of comparing it to the other embedding methods.

Regarding the data provided for the challenge, there were not major limitations associated with it given the extensive pre-processing that was carried out before the DSG week. The only minor issue was the assignment of Curlie labels to domains. These were provided by volunteers of the Curlie project, of which we could not ensure the accuracy.

1.7 Recommendations and Future Work

First of all, we plan to complete a systematic method comparison in future work, and do so at Web scale. Further investigation will also involve research on improved ways of aggregating information from different webpages and methods to embed this into meaningful vector representations. For this latter task, it would be interesting to experiment with adaptations of the GraphSAGE algorithm to this setting, and to make graph-only embeddings scalable via the Local-to-Global approach.

2 Data

The data used during the Data Study Group week comprised four different sampled versions of the Common Crawl web archive[2]. This consisted of open source data, provided by the Common Crawl Foundation with the goal of *"democratising access to web information by producing and maintaining an open repository of web crawl data that is universally accessible and analyzable"*. Overall it contained petabytes of data, collected over 8 years of web crawling, consisting of raw web page data, parsed metadata extracts, and pre-processed webpage content.

For this challenge the datasets were exported from the Common Crawl September 2018 crawl (WET files containing text) and the May-June-July 2018 web-graph. The data extraction pipeline from CommonCrawl to the provided datasets was performed by the challenge organisers in advance, mostly in Spark. This code, as well as a set of starting scripts replicating a simple processing pipeline was provided in a private GitHub repository.

There was a clear need to sample the data prior to the DSG. This was done both in the number of domains included, leading to the four datasets described in Subsection 2.1, and in the information considered per domain. The details of how this second step was performed are to be found in Subsection 2.2.

2.1 Data Description

As already mentioned, for this challenge four different datasets were prepared. These had increasing sizes, so that the performance of methods described in Section 3 could be evaluated at different scales.

Each was divided into three `.csv` files (actually partitioned in several thousands smaller and compressed files in Azure blob storage): a `count` file, a `meta` file, and an `edgelist` file. The information regarding the

[2] `http://www.commoncrawl.org`

Filename	Columns
`count`	`ID` of the domain/node, `count` number of webpages before sampling
`meta`	`ID` of the domain/node, `labels` class used for classification `domain`, `host`, `uri`, `fulltext` of the webpage
`edgelist`	`src` source domain/node ID, `dst` destination domain/node ID

Table 1: Description of the three files provided for each dataset.

columns contained in each file is in Table 1. The rows corresponded to domains in `count`, to webpages in `meta`, and to hyperlinks in `edgelist`.

Datasets contained a list of labelled domains, each with a given number of webpages. The information provided was the text of webpages per domain, and hyperlinks between webpages of domains. This way each dataset corresponded to a multi-digraph[3] with features on its domains/nodes. A list of the datasets, ordered by increasing number of labels, follows. Complete statistics on the number of domains, hyperlinks, labels, and webpages per domain are given in Table 2.

`uk_small_test` Two classes – `Gov` and `Health` – for a total of $2,462$ domains.

`uk_small` Five classes – `Gov`, `News`, `Health`, `Academia`, and `Education` – for a total of $9,260$ domains.

`curlie_single_balanced` A total $46,859$ domains were retained by considering top-level domains only in `["com.", "org.", "uk.", "net.", "au.", "ca.", "edu.", "nz.", "info.", "us.", "gov.", "biz."]`. Domains were given one of seven different labels using information from Curlie[4].

`curlie_single_full` A total $495,966$ domains (filtered as in `curlie_single_balanced`), subdivided in seventeen Curlie classes.

Note that only hyperlinks and webpages text were preserved due to

[3] A graph in which edges have a direction (from source to destination), and multiple edges between the same two nodes are allowed. In our case we could also have self edges, given by a hyperlink from webpage of a domain to another webpage of the same domain.

[4] https://curlie.org/docs/en/about.html

Dataset	No. Domains	No. Hyperlinks	No. Labels	No. Webpages
uk_small_test	2462	34658	2	154858
uk_small	9260	137902	5	554878
curlie_single_balanced	46859	1817190	7	3111485
curlie_single_full	495966	28397947	17	29857208

Table 2: Statistics of the different datasets.

lessons learned from previous work in NCSC that showed that these were by far the most promising sources of features pertinent to a website's function.

Statistics on the graph structure of the four described datasets are available in Section A of the Appendix.

2.2 Sampling

The data were sampled with the following goals:

1. Retain their original characteristics as much as possible;
2. Enable rapid evaluation of new ideas in the DSG;
3. Provide increasing volumes of data so that scaling characteristics of each technique to the full dataset could be extrapolated.

The first level of sampling was carried out on the number of domains and labels considered per dataset, leading to the four datasets described in Subsection 2.1. Then, to further reduce the processing time required by embedding methods presented in Section 3, domains and text information per domain were sampled with the following rules:

1. For every domain/node, we discard it if there are strictly fewer than 10 webpages or more than $1,000,000$: i.e. we discard outliers in terms of very small or very large domains;
2. For every domain/node, we randomly sample 100 webpages of its contents if there are more, or take them all if there are fewer;
3. For every webpage, we consider only the first $2,000$ words, of pages with at least 100 words.

3 Embedding Approaches

This section presents the methods used to generate domain/node embeddings. At first, methods that only consider the textual contents of domains described; we then focused on methods that only take into account the web-graph structure; finally, we look at hybrid methods using both data, such as GraphSAGE.

Embedding methods making use of text information required the generation of aggregated text/features for each domain. In these cases, a variety of strategies were employed, as described in Subsection 3.1.

The feature representations introduced in this Section will be used to classify the domains of the datasets presented in Section 2, using linear classifiers (logistic regression or linear SVC). Results are given in Section 5.

3.1 Text Aggregation

As described in Subsection 2.1, text was provided per webpage. However, text-only embeddings used in this section take a single "document" as input for each domain. As a consequence, all the text belonging to different webpages of a domain needed to be aggregated according to some rule. For this challenge four different text aggregation methods were defined. The first and second aggregated text before the embedding was performed, while the third and fourth relied on aggregating the vectorial representation of domains.

Full text For each domain, concatenate the text of all webpages in that domain, and run an embedding method on this.

Filtered text Using Full text as aggregator, domains may have a different number of words in their aggregated text. To remedy this and speed up the embedding, take the first N words for each domain, where N is the median number of words in each domain. For example on the `uk_small_test` dataset this amounted to (approximately) taking the text of the first 14 pages in each domain, and ignoring the other pages. Since the 100 pages were selected randomly from all pages

in a domain, we would not expect this to cause a bias. Then we run a text embedding method for each domain to extract feature vectors.

Average Run an embedding method over each page individually. The final feature vector will be the average over all webpages in the domain.

Max/Min/Avg Run an embedding method over each page individually. Consider the average feature vector as above. Similarly, the maximum and minimum vectors will be the vectors whose components are the maximum and minimum among all components of vectors of webpages of the same domain. The final feature vector will be the concatenation of the maximum, minimum, and average feature vectors.

During the DSG week there was time to experiment with these text aggregation methods only. Future work on this preprocessing step could involve weighting webpages based on some measure of importance, such as:

1. PageRank calculated within each domain separately;
2. Some heuristic such as length of URI after the host name. For example, `domain.com/about` has 5 characters after the host, and probably has very useful summary features for `domain.com`.

3.2 Text-Only Embeddings

In this section we introduce methods that can be used for documents embeddings. In our case the set of documents associated to a dataset is going to be given either by the the text of individual webpages or by aggregated text at domain level, as described in Subsection 3.1.

3.2.1 doc2vec

This method [10] is based on the word2vec embedding method [6]. It builds features by training a model to predict the next word (in each paragraph), and the topic of each paragraph. For our testing, we built

feature vectors (per page or domain, depending on the aggregation method described above) of length 24.

3.2.2 FastText

FastText [2, 9] is a highly scalable supervised document classification method. However, for this setting, it is also able to output the feature vectors for each document (as well as the linear classifier). We trained FastText to produce 100-dimensional feature vectors (the default) for each domain. To make most use of its scalability, we did not produce feature vectors for each page separately; instead, we ran FastText on the (concatenated) full text of each page in each domain. For the largest dataset (`curlie_single_full`), this was truncated to the first 6000 characters, to reduce the memory requirements. As an example of the difference between FastText supervised and unsupervised modes (using a pre-trained model), Figure 1 shows embeddings obtained in both ways.

3.2.3 ELMo/BERT

ELMo [14] represents the current state of the art in text representations, and produces context-dependent word vectors from a bidirectional language model pre-trained on a large text corpus. A recent evaluation of sentence embedding techniques [13] suggests that ELMo performs substantially better than FastText in downstream NLP tasks. Although the evaluation focussed on sentence representations, it's reasonable to expect some improvement for document-level representations. Another well-known technique we were planning to apply is BERT [5]. Unfortunately, we did not have time to produce ELMo or BERT results during the DSG.

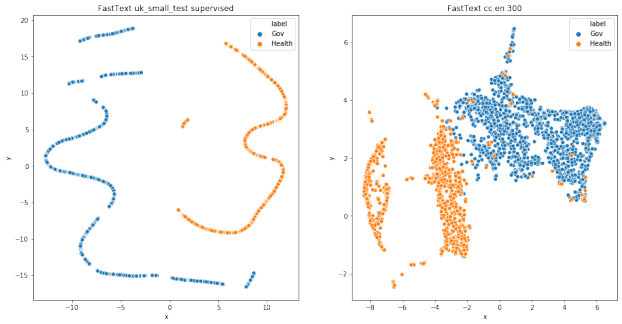


Figure 1: UMAP Dimension-reduction plot of FastText domain vectors for `uk_small_test` when trained using the supervised mode on the left; compared with a pretrained English model on the right.

3.3 Graph-Only Embeddings

In this Subsection we produce embedding using only the hyperlink web-graph of the domains. These provide a second benchmark which can be used for comparison with the content-only embeddings of the previous Subsection and the ones obtained with GraphSAGE in the next one.

We also note that some of these methods do not scale well with increasing amount of data. Hence, it was not possible to obtain embeddings for all the datasets provided during the DSG. This issue motivates the development of the divide and conquer approach of Section 4. With this tool at hand it could be possible to apply node2vec to web-graphs of much greater size. Unfortunately, in the time frame of the DSG, it was not possible to apply this scalability method to the given datasets.

3.3.1 node2vec

This graph embedding technique is based on the skip-gram model, first used for natural language processing [12]. This assumes that words found in similar contexts have similar representations, and that the “best” representation should optimize the likelihood function over a neighborhood of words. We treat graphs as documents and employ

substructures and neighbourhoods defined by node2vec as words composing them. The process relies on sampling a biased random walk starting at a given node as described in [7].

3.3.2 Graph Convolutional Networks

The graph structure is used as input to a neural network to produce embeddings. In particular, during the Data Study Group week we were able to test the effectiveness of Undirected GCN [3, 15] and Gated GCN [11] on the classification task. The inputs for the first method were the number of triangles in the graph, and the degree of nodes. In a similar way, the second method employed number of triangles, in degrees and out degrees of nodes.

3.4 Hybrid Embeddings

The main objective here is to show that the text and graph structure information can be combined to obtain embeddings, which can outperform embeddings using only one type of information.

3.4.1 GraphSAGE

The chosen method was GraphSAGE [8]. Its input is a graph with feature vectors on its nodes. The algorithm will then aggregate features of neighbours of a node to provide its embedding as a vector in some $\mathbb{R}^d$. To ensure scalability it does not consider all neighbours. Instead, a sample of these is obtained via a random walk, starting at the node which needs to be embedded. In our setting features of nodes were the above mentioned doc2vec emebeddings, and the graph structure was the one provided by hyperlinks. The three steps of the method are illustrated in Figure 2.

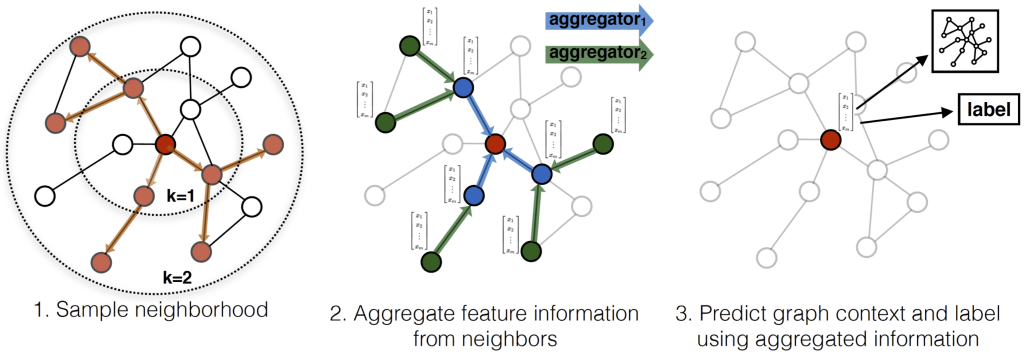


Figure 2: The GraphSAGE method.

4 Scalable Embeddings

The scalability of graph and hybrid embedding methods described in the previous sections is of paramount importance due to the vast size of the world wide web. Therefore, in this Section we present a divide and conquer technique to make any embedding method highly scalable. This method utilizes the local information of each node in the network to create effective local embedding as a first step. Then, such local embeddings are synchronised – scaled, rotated, and translated – to form a global embedding. This was tested on synthetic data only, and we leave as future work its application to real-world datasets.

4.1 Description of the Method

We name our method Local-to-Global (LTG), an extension of the As-Synchronized-As-Possible (ASAP) [4] for embeddings of arbitrary dimensions. The input of the method is a graph embedding method together with a graph. It could also be used in combination with hybrid methods (as GraphSAGE), but in that case feature vectors of nodes of the given graph would be required as inputs as well.

LTG first clusters the graph into overlapping subgraphs, here called patches. Each patch is then embedded on its own, using an algorithm of choice (node2vec or GraphSAGE for example), the results are called local embeddings. In the end, in order to produce a global embedding we proceeded as follows.

1. The local embeddings are first brought to the same scale by approximately matching the distances between embedded nodes belonging to overlapping parts of the original patches.
2. Local embeddings are then rotated (or reflected), to optimally align them in a globally consistent way.
3. The final embedding position of each node is found by solving a least square problem, which results into an over-determined linear system.

For more details on the method see the Annex file to this report: “A scalable divide-and-conquer approach for network embeddings”.

4.2 Results on Synthetic Data

We successfully implemented the LTG method testing it on artificially generated patches. These were the result of the following procedure:

1. We randomly sampled a set of points in the plane from one (or more) Gaussian distributions;
2. Among all the points obtained during the first step we subsampled a few, which we called landmark points;
3. Finally we split all the points into subsets based on the Voronoi diagram of the landmark points. These will be our patches. Moreover, Voronoi regions were enlarged by a constant ϵ, that could be used as a parameter regulating the level of intersection between patches.

The code to produce the artificial data for the extended version of ASAP is available in the challenge Github repository.

Figure 3 illustrates the LTG algorithm on synthetic data, where the different patches were obtained by randomly scaling, rotating, and translating the synthetic ones.

Discussion. By combining LTG with powerful but non-scalable emebedding methods, we would be able to obtain global embeddings of graphs which are now out of our range. The results obtained during the DSG week are very promising. We also started experimenting with the required patches creation step (using spectral clustering [16] and graph modularity maximization via the Louvain method [1]), but unfortunately there was no time to implement the method on the provided datasets in order to obtain embeddings for classification. We suggest this as a future direction of work.

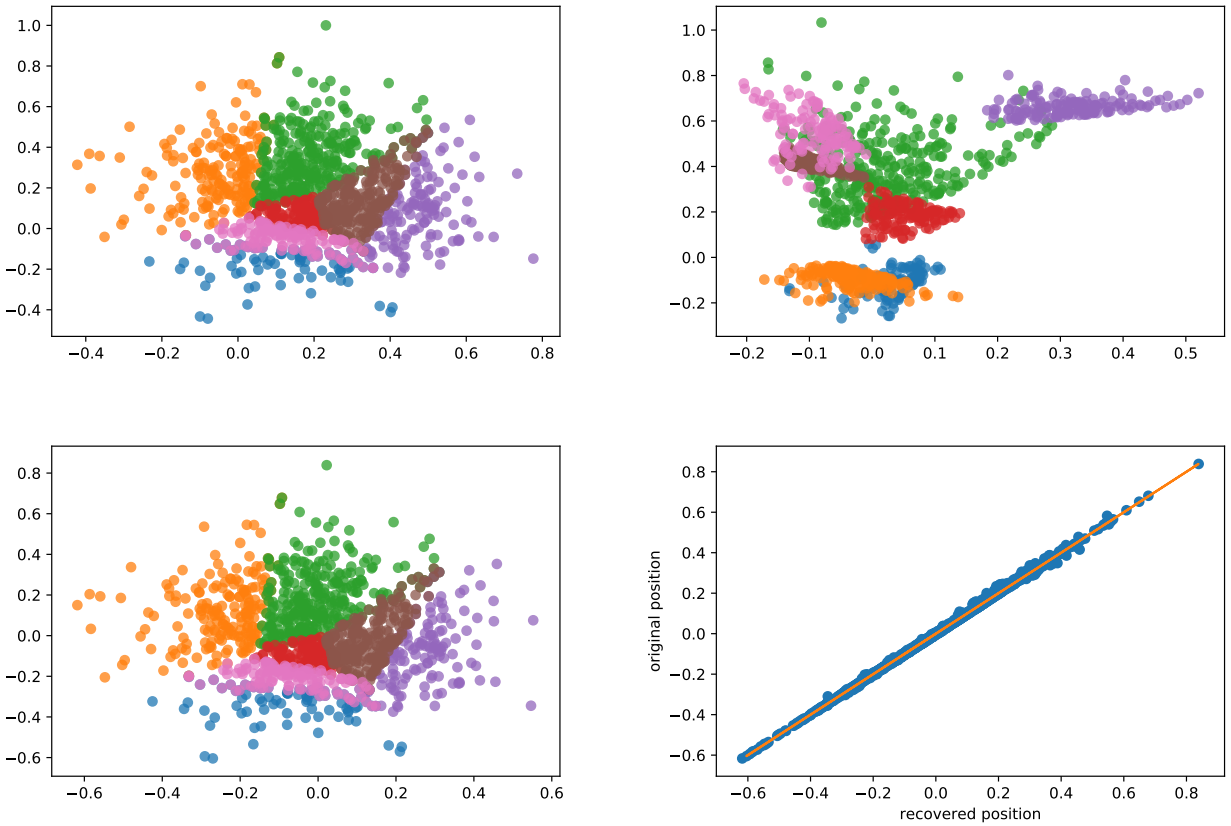
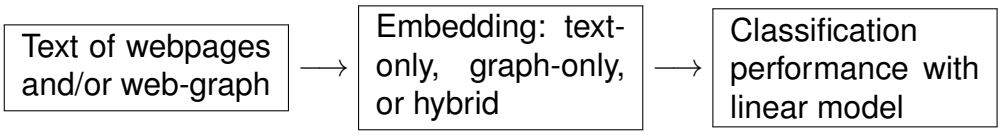


Figure 3: Illustration of the Local-to-Global algorithm. The top row shows the synthetic data on the left and the randomly perturbed patches on the right. The bottom row shows the reconstruction of the data as given by the LTG algorithm on the left, and the scatter plot of the original coordinates versus the recovered ones on the right.

5 Results

Here we compare the various features derived from methods described in Section 3. The goal was to find the features with best classification performance on the four given datasets. The pipeline of our analysis can be summarized by the following diagram.



In the middle step, text-only methods relied on some aggregation that could be performed either before or after the embedding (see Subsection 3.1). Also note that the linear classifier used was a linear SVC for text-only embeddings, and logistic regression in all other cases.

In general, we were able to apply each embedding method only to a subset of the datasets. Further experimentation is needed in this direction to check the stability of methods at different scales. Morevoer, we remark that further robustness checks should be run on hyperparameters used in embedding methods and on the different steps of the pipeline in general.

5.1 Text-only Embeddings

A comparison of different text embedding methods is given in Table 3. We obtained accuracy rates for the domain classification problem, using a linear SVC with a 10% random test set and no cross-validation. It should be kept in mind that since FastText is trained using domain labels, and uses larger feature vectors than doc2vec, the results presented here are not directly comparable. Also, further tests should be run on the optimal number of features to be used in this setting.

For doc2vec, calculating per-page feature vectors for the `curlie_single_full`, we needed to reduce the size by randomly selecting

15% of the webpages in each domain (using the full set of domains). We expected this not to significantly affect the results — using 15% page sampling for `uk_small` produced a similar accuracy rate to using all the pages.

We also note that these results all make use of linear classifiers. Better results can be achieved with different models; for instance, a random forest with doc2vec (averaging) embeddings achieved 96.9% test accuracy on `curlie_single_full`, a substantial improvement.

Dataset	Aggregation	Embedding Method	No. features	Test accuracy
`uk_small_test`	Filtered Text	doc2vec	24	0.943
	Average	doc2vec	24	0.984
	Max/Min/Avg	doc2vec	72	0.980
	Full text	FastText	100	**1.000**
`uk_small`	Average	doc2vec	24	0.981
	Max/Min/Avg	doc2vec	72	**0.985**
`curlie_single_balanced`	Full text	Fast text	100	**0.921**
`curlie_single_full`	Average	doc2vec	24	**0.750**
	Full text	FastText	100	0.722*

Table 3: Top-1 test accuracy score obtained with linear SVC and no cross validation. The * FastText results were based on using the first $6,000$ characters of text from every domain.

Discussion. From Table 3, it is possible to see that for doc2vec, the averaging aggregation method seems to provide the best results: it improves notably over the filtered text approach, and performs similarly to the Max/Min/Average method (but with much fewer degrees of freedom). We cannot directly compare with FastText, but both produce good results and could be used in the future. However, we note that FastText requires labelled domains and trains using these labels, so testing on other tasks would be an important verification step.

On the largest test set (`curlie_single_full`) doc2vec and FastText perform similarly using linear SVCs. Unfortunately, results are not directly comparable due to the different number of features used and the different sampling assumptions made.

Example: Domain discovery. To demonstrate the success of the text embeddings, we considered some of the nearest neighbours produced by doc2vec with averaging aggregation (from `uk_small`).

For the Alan Turing Institute (`turing.ac.uk`), the five nearest neighbours (with descriptions taken from these sites) were:

1. Science and Engineering South Consortium (`ses.ac.uk`);
2. Imperial College London (`imperial.ac.uk`);
3. Whole Systems Energy Modelling Consortium (wholeSEM), a groundbreaking multi-institution initiative to develop, integrate and apply state-of-the-art energy models (`wholesem.ac.uk`);
4. University of Southampton (`southampton.ac.uk`);
5. Software Sustainability Institute, an institute that cultivates better, more sustainable, research software to enable world-class research (`software.ac.uk`).

For the NCSC (`ncsc.gov.uk`), the five nearest neighbours were:

1. `jiscdigitalmedia.ac.uk`, which redirects[5] to `jisc.ac.uk`. Jisc provides digital solutions for UK education and research;
2. `netskills.ac.uk`, which redirects to `jisc.ac.uk`;
3. Cyber Aware, a cross-government awareness and behaviour change campaign delivered by the Home Office in conjunction with Department of Culture, Media & Sport alongside the National Cyber Security Centre, and funded by the National Cyber Security Programme in the Cabinet Office (`cyberaware.gov.uk`);
4. Centre for the Protection of National Infrastructure, a government authority for protective security advice to the UK national infrastructure (`cpni.gov.uk`);
5. `jiscinfonet.ac.uk`, which redirects to `jisc.ac.uk`.

Both of these lists demonstrate successful domain discovery: All neighbours are similar to the original domain in either function

[5] Indicating that the dataset is no longer completely current.

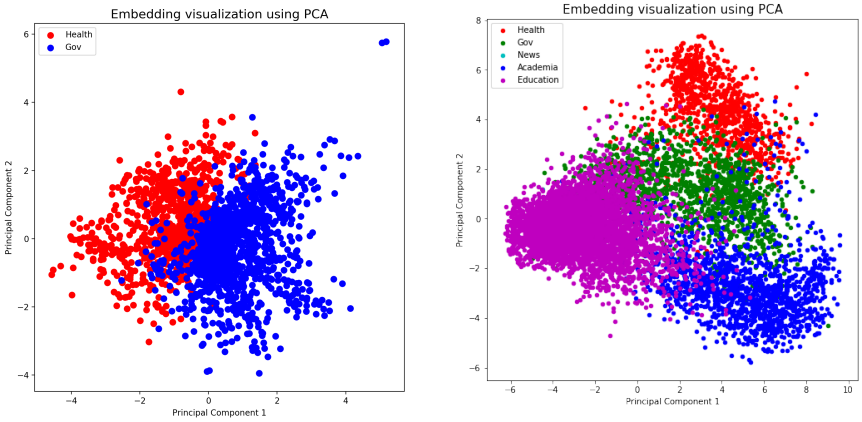


Figure 4: node2vec features in projected in $\mathbb{R}^2$ with PCA of `uk_small_test` and `uk_small` respectively.

(Universities similar to the Turing Institute) or in structure (NCSC and CPNI both deal with public security issues, one digital security and one physical security).

5.2 Graph-only Embeddings

On `uk_small_test` and `uk_small` node2vec performed much better than Graph Convolutional Networks. Table 4 reports test accuracies and $F1$ scores obtained with logistic regression and the feature vectors of the two embedding methods. In the case of node2vec it was not possible to provide embeddings of the biggest datasets due to scalability problems. On the other hand GCN could do so, but provided poor performance.

The former of the two methods was run setting $p = 1$, $q = 0.5$, and fixing dimension $k = 10$ for `uk_small_test` and $k = 100$ for `uk_small`. A plot of the obtained features is given in Figure 4, where the dimension was reduced using Principal Component Analysis (PCA).

Dataset	Embedding Method	Test accuracy	F1 score
uk_small_test	node2vec	**0.97**	**0.97**
	Undirected GCN	0.725	0.722
	Gated GCN	0.563	0.541
uk_small	node2vec	**0.95**	**0.95**
	Undirected GCN	0.728	0.674
curlie_single_balanced	Undirected GCN	**0.138**	**0.052**

Table 4: Logistic regression test accuracy and F1 scores produced by graph-only embedding methods.

Discussion. Graph-only embedding provided good results for small datasets. Unfortunately, node2vec is not able to scale to very big graphs in its present form. As a result, a future direction of work that could solve this problem could be the combination of node2vec with the Local-to-Global approaches described in Subsection 4.

5.3 Hybrid Embeddings

We were able to test the performance of GraphSAGE on uk_small_test on the label classification task via logistic regression. Two sets of features were used to initialize the method, and results are presented in Table 5 and Table 6 respectively. These were both derived from doc2vec embeddings; in the first case we used the doc2vec feature vectors of filtered text, while in the second case we used the average doc2vec feature vectors (see Subsection 3.1).

All hyperparameters in GraphSAGE were set to the default values of the reference TensorFlow implementation.[6] Training was carried out for a total of 10 epochs, as running for additional epochs did not result in any notable performance gains. We compared the performance of the doc2vec features already described, with the performance of features obtained by running GraphSAGE using these as inputs. Moreover, three methods – Mean, LSTM, and Maxpool – were implemented for the GraphSAGE aggregation step.

[6]See https://github.com/williamleif/GraphSAGE.

Using doc2vec features of filtered text

Method	Micro-F1 score
Without GraphSAGE	0.9378
GraphSAGE with Mean aggregator	0.9756
GraphSAGE with LSTM aggregator	0.9782
GraphSAGE with Maxpool aggregator	**0.9919**

Table 5: Classification of `uk_small_test`, with and without GraphSAGE, using logistic regression and doc2vec embeddings of the filtered text of each domain.

Using average doc2vec features

Method	Micro-F1 score
Without GraphSAGE	0.9858
GraphSAGE with Mean aggregator	0.9919
GraphSAGE with LSTM aggregator	**0.9959**
GraphSAGE with Maxpool aggregator	0.9878

Table 6: Classification of `uk_small_test`, with and without GraphSAGE, using logistic regression and the average doc2vec embedding of each domain.

Discussion. It was only possible to apply the GraphSage embedding method to the `uk_small_test` dataset. The results obtained provided initial evidence that GraphSAGE is able to improve on the performance of text-only embeddings. The avenues that still need to be explored include a better way of random sampling neighbours of a node, and improved aggregators.

6 Conclusions

Given the three different approaches we implemented for the label classification task, results can be summarized as follows.

- Text and graph embeddings alone can perform well on the label prediction problem. In particular, the best results are provided by text-only embeddings which also do not suffer the scalability problems involved with graph-only approaches.
- There is some initial evidence that GraphSAGE can outperform text-only embeddings. This was shown on `uk_small_test`, where GraphSAGE was able to improve the performance of the given doc2vec embedddings of filtered text. We also note that this method shows good scalability properties.
- It was possible to obtain 96.9% test accuracy in classifying `curlie_single_full` labels with doc2vec and a random forest (this was done by randomly sampling 15% of the web-pages per domain to reduce the computational and memory cost). This suggests that text-only embedding could be "good enough" for the label classification problem.

Regarding scalability methods described in Section 4 it was given a proof-of-concept on artificial data. This should allow the embedding of much larger networks, especially with more expensive embedding methods, and might even enable representations to be created for the 1B+ individual website hosts.

In conclusion, the results obtained were very promising and have the potential to pioneer new methods which will enable significant advances in the discovery and understanding of the web.

7 Future work and research directions

This work has demonstrated the potential to create a new way of representing the web through vector embeddings of domains. However, this is in its very early days and much work remains to be completed and new research avenues investigated. This section briefly describes some of these.

7.1 Building to Web-Scale

One of the most important items to consider, for future work, is how to sample content when the project is scaled up. Currently for each pay-level domain, 100 pages are chosen uniformly at random and their content is added to form the features for the domain. However, a large number of websites contain over $10,000$ pages and thus sampling pages uniformly at random are:

1. Unlikely to select the pages that users visit regularly;
2. Unlikely to select the pages that contain words most likely to help with classification.

Thus another method for generating feature vectors is required. A few examples which could be considered are:

- Choose the 100 pages that are most frequently visited by users (which we could perhaps approximate through an intra-site page-rank for each page);
- Choose the 100 pages whose word vectors have the most discrimination power (through computation of feature importance, perhaps through a random forest classifier);
- Choose the 100 pages with the most words;
- Use clustering methods to pick the “most representitive” pages (perhaps by embedding each page separately, clustering the embeddings, and sampling only from the largest cluster).

7.2 Further Experiments

Text-only Further investigation of BERT/ELMo is needed to understand whether GPU acceleration or use of the Local-to-Global technique could be used to speed up processing enough to be able to process the full dataset. There was not enough time during the DSG to process even the Curlie balanced dataset with a naive approach.

node2vec Due to scalability problems we were not able to compute embeddings of the `curlie_single_balanced` and `curlie_single_full`. It would be interesting to apply the Local-to-Global method in this setting.

GraphSAGE There were issues that needed resolving before representative results for the two largest datasets could be given. For the balanced dataset, this should just be parameter optimisation. All experiments showed that it is the parameter selection that is impacting on the accuracy. Thus a day or so of experiments should resolve this. The first parameter worth focusing on is the size of the neighbourhood sampled to update the representation. The larger dataset may be more problematic. Initial experiments appeared to show that the current implementation of GraphSAGE is not suited to such a large dataset. Thus a new implementation might be required to process this dataset and to perform the task of labelling the unlabelled portion of the Curlie dataset.

Another set of experiments left for future work should focus on different tasks/objectives, such as link prediction, which we did not cover during the DSG.

7.3 GraphSAGE-based Research Avenues

Part of the appeal of GraphSAGE, is its ability to work with any graph, where each node can contain features in a fairly general form. However, while this generality can be seen as a strength, it does mean that it is not finely tuned to any particular task. As future work, we propose to adjust GraphSAGE to the web-graph setting by considering the (normalised)

PageRank scores of domains. This would provide a measure of the importance of the different nodes, and could be used to implement the random sampling and aggregation steps with weights reflecting this.

Also, while the problem we have considered in this work was a supervised one, the labelling of Curlie domains is actually a semi-supervised one. Only a small portion of this dataset has been labelled, and the final goal is to label to remaining part. We conjecture that GraphSAGE could be adjusted to handle this situation, and suggest this as another direction of future research.

7.4 Further Extension of the Local-to-Global Approach

The Local-to-Global (LTG) approach presented in Subsection 4 has the potential of scaling up embedding algorithms of high accuracy to inaccessible graph sizes. While in this report we have just tested the algorithm on synthetic data, it is essential to further test this approach on real-world datasets. This would also require a choice for the patches creation step; some possible solutions could be:

1. Given some text-only embedding split the data in Voronoi regions as done with the synthetic datasets used to test the Local-to-Global algorithm. Patches can be extended so to intersect pairwise (as required by LTG) by adding some or all of their 1-hop neighbours, i.e. nodes connected by an edge to some node in the patch;
2. Spectral clustering, followed again by the addition of some or all 1-hop neighbours as above.

Finally, further investigation is needed on a variety of accurate - but computationally expensive - graph embedding algorithms, to fully exploit the power of this divide and conquer strategy.

References

[1] Vincent Blondel et al. “Fast unfolding of communities in large networks”. In: *arXiv: 0803.0476* (2008).

[2] Piotr Bojanowski et al. “Enriching Word Vectors with Subword Information”. In: *Transactions of the Association for Computational Linguistics* 5 (2017), pp. 135–146.

[3] Benedikt Bünz and Matthew Lamm. “Graph Neural Networks and Boolean Satisfiability”. In: *arXiv: 1702.03592* (2017).

[4] Mihai Cucuringu, Amit Singer, and David Cowburn. “Eigenvector synchronization, graph rigidity and the molecule problem”. In: *Information and Inference: A Journal of the IMA* 1.1 (2012), pp. 21–67.

[5] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *arXiv: 1810.04805* (2018).

[6] Yoav Goldberg and Omer Levy. “word2vec Explained: deriving Mikolov et al.’s negative-sampling word-embedding method”. In: *arXiv: 1402.3722* (2014).

[7] Aditya Grover and Jure Leskovec. “node2vec: Scalable feature learning for networks”. In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2016, pp. 855–864.

[8] Will Hamilton, Zhitao Ying, and Jure Leskovec. “Inductive representation learning on large graphs”. In: *Advances in Neural Information Processing Systems* (2017), pp. 1024–1034.

[9] Armand Joulin et al. “Bag of Tricks for Efficient Text Classification”. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers* (2017), pp. 427–431.

[10] Quoc Le and Tomas Mikolov. “Distributed representations of sentences and documents”. In: *ICML’14 Proceedings of the 31st International Conference on International Conference on Machine Learning* (2014), pp. II-1188–II-1196.

[11] Yujia Li et al. “Gated Graph Sequence Neural Networks”. In: *arXiv: 1511.05493* (2015).

[12] Tomas Mikolov et al. “Efficient estimation of word representations in vector space”. In: *arXiv: 1301.3781* (2013).
[13] Christian S. Perone, Roberto Silveira, and Thomas S. Paula. “Evaluation of sentence embeddings in downstream and linguistic probing tasks”. In: *arXiv: 1806.06259* (2018).
[14] Matthew E. Peters et al. “Deep contextualized word representations”. In: *arXiv: 1802.05365* (2018).
[15] Franco Scarselli et al. “The graph neural network model”. In: *IEEE transactions on neural networks* 20.1 (2009), pp. 61–80.
[16] Ulrike Von Luxburg. “A tutorial on spectral clustering”. In: *Statistics and computing* 17.4 (2007), pp. 395–416.

Appendix

A Data Analysis

We present some additional information regarding in and out degrees and the number of connected components of the hyperlink graphs of the datasets under study.
This information is provided in Table 7 and Table 8 respectively.

Dataset	Density	In degree				Out degree			
		Min	Median	Mean	Max	Min	Median	Mean	Max
uk_small_test	0.0057	0	4	14	827	0	7	14	511
uk_small	0.0016	0	2	15	4935	0	5	15	4590
curlie_single_balanced	0.00083	0	4	39	36047	0	4	39	36440
curlie_single_full	0.00011	0	7	57	364207	0	9	57	340468

Table 7: Statistics about degree distribution of in and out edges.

Dataset	No. conn. comp. (weak)	No. conn. comp. (strong)	Size giant
uk_small_test	29	467	2434
uk_small	88	1090	9172
curlie_single_balanced	1060	8182	45799
curlie_single_full	4657	48126	491302

Table 8: Statistics about connected components in all datasets.

Also, we consider uk_small_test in particular and provide information regarding number of nodes per class, the effect of sampling on the number of web-pages, and the degree distributions of in and out nodes.

Each node in the hyperlink graph is either labelled as Gov or Health. The number of nodes per class are given in Table 9.

Class	Number of nodes
Gov	1476
Health	987

Table 9: Nodes per class.

We look at the third sampling assumption on domains by comparing the distribution of the number of webpages before and after sampling. Figure 5

shows that for `uk_small_test` the distribution after sampling changes from a power law distribution with $(P(x) \sim x^{-\beta})$ where $\beta = 1.51$ to a distribution with a peak at 100. This observation applies to all datasets, the peak at 100 becoming more substantial as the size of the dataset grows.

The other assumptions will require further investigations by comparison with the full datasets, before sampling, which were not provided for the DSG.

Finally, a plot of the node degree distribution of the hyperlink graph is presented for `uk_small_test` in Figure 6, where the dashed line represents a power law fit of $(P(x) \sim x^{-\beta})$ with $\beta = 1.61$. Moreover, the distributions of in and out degrees for the same web-graph are presented in Figure 7.

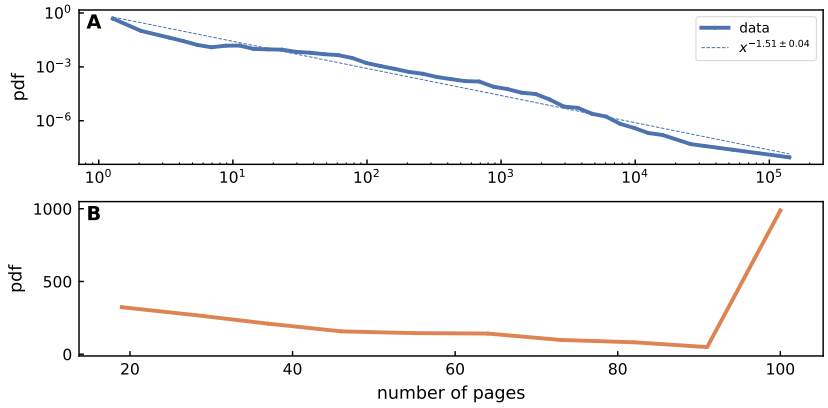


Figure 5: Distribution of the number of web-pages before **(A)** and after **(B)** sampling.

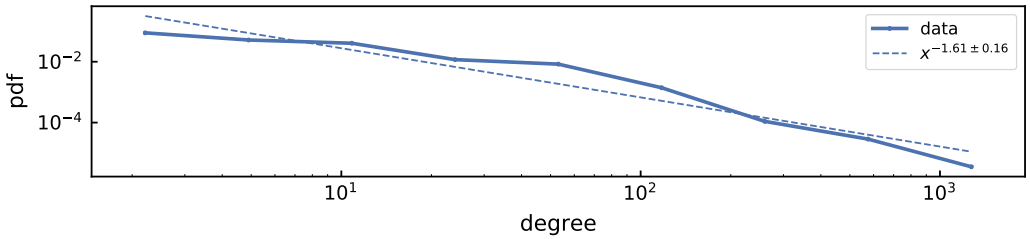


Figure 6: Degree distribution of hyperlink graph of `uk_small_test`.

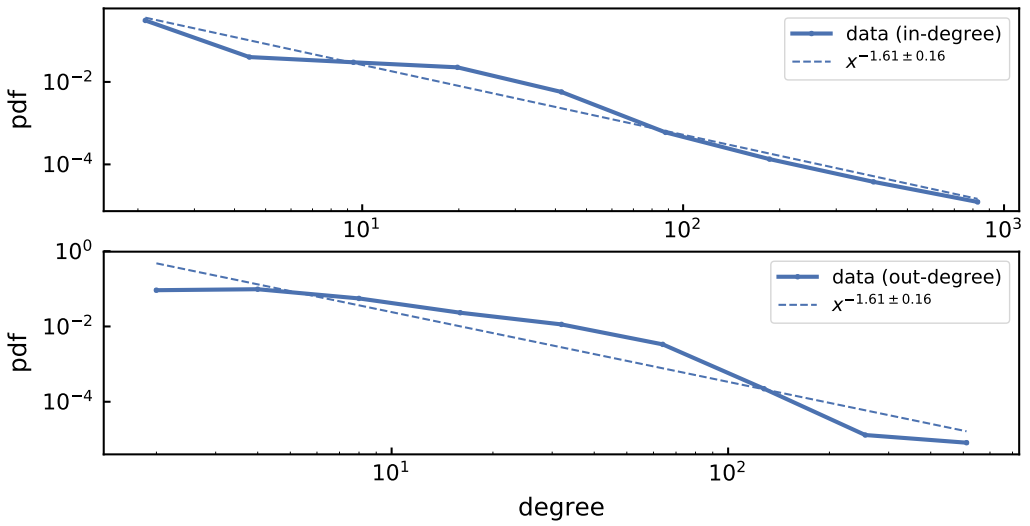


Figure 7: Degree distribution of hyperlink graph of `uk_small_test` considering in and out edges separately.

B Team members

In alphabetical order by first name:

Abeer ElBahrawy is a doctoral student at the University of London and a Turing Enrichment student. She performed part of the descriptive analysis.

Aldo Glielmo is a doctoral student at King's College London, interested in the many potential applications of data science to physics. He worked on the implementation and testing of the ASAP algorithm of the Local-to-Global approach.

Benjamin Sach is a Data Scientist with The Alan Turing Institute. He likes problems at the intersection of mathematics and computer science. At the DSG he computed FastText word embeddings and classifications for the `curlie_single_full` dataset.

Daniel Martin is a Research Fellow at the University of Bristol. During the data study group he worked on the GraphSAGE embeddings, tailoring the algorithm for website data.

Gabriele Beltramo is a doctoral student at Queen Mary University and a Turing Enrichment student. He worked on the synthetic data generation and the Local-to-Global approach.

Giovanni Colavizza is a senior research data scientist at The Alan Turing Institute. He works on the application of machine learning and data science to foster scientific research, with a focus on the humanities and social sciences. He is the challenge facilitator for this group.

Lindon Roberts is a doctoral student at the Mathematical Institute, University of Oxford. He contributed to this project by implementing and analysing the text-only embeddings.

Lucas Deecke is a doctoral student at the University of Edinburgh working on the intersection of probabilistic machine learning and deep learning. He contributed an analysis of GraphSAGE, and evaluated its susceptibility to different document embeddings.

Marya Bazzi is a Turing Research Fellow. Before joining the Turing Marya

was a postdoctoral scholar at Oxford University and Head of Analytics at a London-based FinTech. Marya is the science lead for the project.

Mihai Cucuringu is an Associate Professor at University of Oxford, and has had a fellowship at The Alan Turing Institute since 2017. His research interests span machine learning, networks, and time series analysis. He was a Science Lead for the project, and his contribution amounted to proposing a Local-to-Global approach.

Mridul Seth is a doctoral student at UCLouvain, Belgium. He worked on implementation of Local-to-Global embeddings.

Nikolas Kuhlen is a doctoral student at the Alan Turing Institute with supervisors from the universities of Warwick and Oxford. He worked on the text embeddings, in particular LDA.

Paul Jones is a data science lead at the National Cyber Security Centre, and a visiting researcher under the Alan Turing Institute's Defence and Security program. He is the challenge owner for this group.

Prateek Gupta is a DPhil student at the University of Oxford. His research is focused on using machine learning methods to solve NP-Hard problems. He worked on checking the feasibility of graph convolution networks for node classification on big graphs.

Spiros Denaxas is a biomedical researcher at the Institute of Health Informatics, University College London. His contribution was the implementing of the pipeline to evaluate text-only embeddings.

Yiliu Wang is a first year Statistics PhD student at London School of Economics and Political Science. Her contribution at the DSG is the implementation and evaluation the node2vec embeddings.

Yonatan (Yoni) Dukler Is a second year PhD student in applied mathematics from UCLA. He contributed to the Local-to-Global approach by implementing the ASAP algorithm.

turing.ac.uk
@turinginst