LSDE Group 16 Project Plan

W1: Covid-19 on the Web

Lu Wang
Student nr: 2754874
l.wang2@student.vu.nl

Weiqiang Guo
Student nr: 2748901
w.guo2@student.vu.nl

Zheng Zhang
Student nr: 2748843
z.zhang10@student.vu.nl

1. MOTIVATION AND DESCRIPTION

1.1 Motivation

From 31 December 2019, when the Wuhan Municipal Health Commission notified the pneumonia outbreak, to 26 September 2022, it has been exactly 1,000 days since the coronavirus outbreak entered the public consciousness. Over the past 1,000 days, the outbreak has had a profound impact on people's lives and economic activities in various countries. People have started to adapt to wearing masks for travel and gradually removed them, and we have become accustomed to regular nucleic acid testing. The length and depth of the impact of the coronavirus was unimaginable at the beginning of the outbreak.

With the aid of web crawl data provided by Common Crawl, we think it is worthwhile to review the dominant theories and trending topics and how they changed over time in each period during the preceding three years. These details are a representation of how British people, negotiate with, and are influenced by the corona virus. The findings of our work should provide users with an overview of how Covid-19's effects British people during the last three years.

1.2 Description of project

The objective of this project is to process and analyze three forms of crawl data from Common Crawl to extract websites from the UK that are linked to Covid-19 and to create a timeline where each time spot provides a summary of information, including key terms and the most significant sentence. The three types of data include, WARC: which stores the raw crawl data, WAT: computational metadata for the data stored in WARC WET:extracted plaintext from the data stored in the WARC

Due to the enormous amount(several hundred TBs a month) of data provided by Common Crawl and the limited computer power and financial resources we have, we must successfully complete two steps of data selection in order to complete this project. Since we are only interested in UK information, the first step is to extract information only from UK web pages. The second step uses keywords to select information directly related to covid-19 web pages. The amount of extraneous data can be significantly decreased through the two-step data selection, which allows us to have the data we can work on and commence our research.

For the analysis work, we have two parallel workflows which demonstrates in Figure 1. The first one uses WAT data from the UK to perform PageRank to rank the importance of the websites and use the topology map to show the citation networks of the websites. The second one use text data extracted from UK WET data, Spark NLP for sentiment analysis to visualise how British people think about Covid19. Apart from that, we also combine the outcomes of the first and the second part to give more accurate results. We use PageRank measure the importance of a web page. In addition, we also take the relevance of a web page to Covid-19 into account. With combination of the importance and relevance, we get a composite score to measure the importance of a page. This socre(or weight) can help us do sentiment analysis more accurately. The more a web page is, the emotions on that page are also more important.

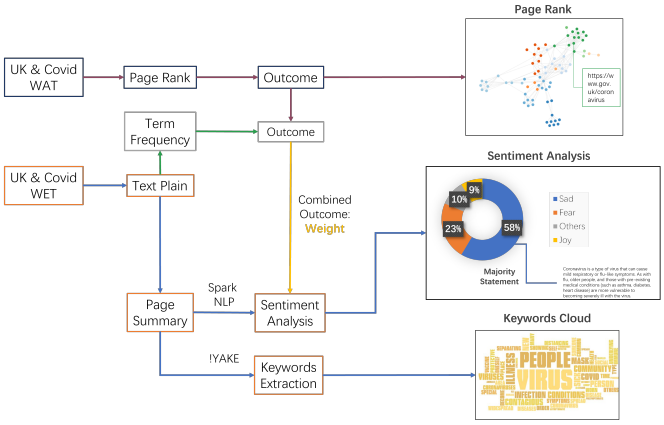

Figure 1: Project Outline

2. INTRODUCTION OF DATA

In this section, we briefly introduce our dataset, the tools used to read it, and some of the distribution characteristics of the dataset.

2.1 Data Format and Tools to read it

The Common Crawl corpus contains petabytes of data collected via automatic web crawling since 2008. It contains raw web page data, extracted metadata, and plain text. It has three main file formats which are WARC, WAT and WET.

The WARC format is raw data from the crawl. It stores the HTTP response, information about how that information was requested, and metadata on the crawl process itself. The WAT and WET formats are the summarization of the

WARC format. The WAT format stores important metadata about the records stored in the WARC format. The WAT format is about a third the size of the WARC format. By contrast, the WET format only stores extracted plain text. The WET format is about one-sixth the size of the WARC format.

There is another important data in the common crawl corpus. It is the common crawl index, which stores the indexes to WARC files. With the help of the index, we can access WARC files quickly.

In our project, we plan to analyze the WAT and WET files rather than the WARC files. There are two reasons to explain why. First, the WAT and WET files contain all data we need, including the plain text of the web pages, and links in the web pages. Second, WAT and WET files are much smaller than WARC, which means that it is much easier to handle.

To read the WAT and WET files, we plan to use the WARCIO library[1] which provides simple methods to parse WARC files, WAT files and WET files.

2.2 Data Size

In our project, we would like to know how the British people's perception of Covid-19 has changed over time. Hence, we selected WARC files ranging from January 2020 to August 2022. Apart from that, we only consider the web pages whose URL contains the Covid-19 keyword. After filtering WARC files, we get about 82GB WARC data(note that it is a very rough estimation). It is much smaller than before, which can be easier to analyze.

WAT and WET files are much smaller than WARC files. It is much better that we can analyze WAT and WET files instead of WARC files since there is less data, but it still contains the data we need. The volume size of them is about half of WARC files, which is about 41GB.

2.3 Data Distribution

We are also wondering how many web pages are related to Covid-19 at a given time period in UK. The URL count trend is shown in Figure 2,

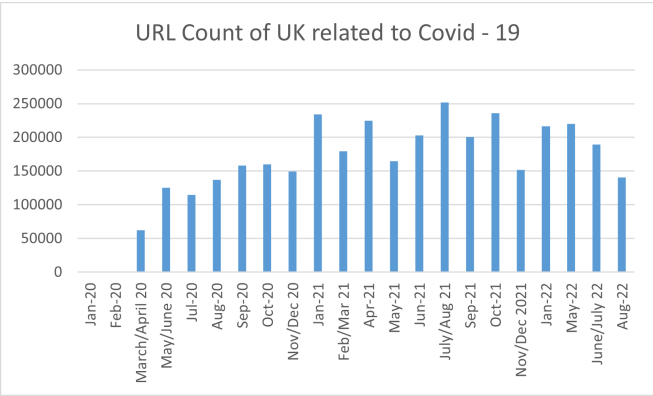

Figure 2: The number of URLs related to Covid-19 in UK

where the horizontal axis of the image is the date, from January 2020 to August 22. Note that some data are for two months rather than one. This is because common crawl sometimes crawled two months of data at a time. The vertical axis is the number of URLs related to Covid.

Figure 2 shows the trend exactly as we would expect. Little attention was paid to Covid-19 when it first appeared, but as Covid-19 grew in severity, attention to it increased and has remained at a high level since then. There is a noticeable decreasing trend of

3. RELATED ALGORITHMS AND PACKAGES

In this section, we briefly introduce what algorithm we would use in our project and corresponding libraries.

3.1 PageRank Algorithm and GraphDataframe

3.1.1 PageRank Algorithm

PageRank [7] is an algorithm that is commonly used by search engines to rank web pages. It turns out to be an efficient algorithm. It is able to measure the importance of website pages by considering the number and quality of links to a page. In short, the more links to a site, and the better the quality of links, the more important the site is.

PageRank uses PR(PageRank) value to measure the importance of a website page. The higher the PR value is, the more important the corresponding website is. Assume that we have a web page A, its PR value can be calculated by the following equation,

$$PR(A) = \frac{(1-d)}{N} + d(\frac{PR(T_1)}{C(T_1)} + \cdots + \frac{PR(T_n)}{C(T_n)}) \quad (1)$$

where T_i is a page that links to page A, $C(T_i)$ is the number of outbound links of Page T_i, d is the damping factor that measures the probability that at any given moment, a user arrives at a page and continues to navigate backward, N is the total number of web pages.

In fact, PageRank values can be viewed as the entities of right eigenvectors of an adjacency matrix. The eigenvectors can be expressed by the following equation,

$$R = \begin{bmatrix} PR(p_1) \\ PR(p_2) \\ \cdots \\ PR(p_n) \end{bmatrix} \quad (2)$$

where R is the solution of the following equation

$$R = \begin{bmatrix} (1-d)/N \\ (1-d)/N \\ \cdots \\ (1-d)/N \end{bmatrix} + d \begin{bmatrix} l(p_1,p_1), l(p_1,p_2), \cdots l(p_1,p_n) \\ l(p_2,p_1), l(p_2,p_2), \cdots l(p_2,p_n) \\ \cdots \\ l(p_n,p_1), l(p_n,p_2), \cdots l(p_n,p_n) \end{bmatrix} R \quad (3)$$

where $l(p_i, p_j)$ is the ratio between the number of links from page p_i to page p_j and the total number of outbound links of page p_j.

This equation can be solved by the power iterative method or the power method. After we get the solution of Equation 3, we get the PR value of every page, and then we can rank the web pages according to their PR values.

In our project, we plan to use the PageRank algorithm to measure the importance of web pages that we care about.

3.1.2 GraphFrame

GraphFrame [3] is a package that provides DataFrame-based Graphs. It not only provides the functionality of

GraphX [10], but also some new functionality based on Spark DataFrames. We plan to use this package to implement the PageRank algorithm.

3.2 TF-IDF Algorithm

Apart from the importance of the web pages, we are also wondering about the relevance of the web pages to our keywords. For example, we are curious about how relevant each page is to the keyword Covid-19. We first introduced TF-IDF algorithm which inspire us to come up with a way to measure relevance in our project,

The term frequency-inverse document frequency(TF-IDF) algorithm[9] is used to measure how relevant a user query is to a document in a collection of documents. In our project, a document is a web page. The degree of relevance is determined by the tf-idf value. The tf-idf value is determined by two statistic metrics. One is term frequency(tf), and the other is inverse document frequency(idf).

The term frequency is intended to measure the relative frequency of a term within a document. It is defined by the following equation,

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{4}$$

where t represents the term, d represents the document, $f_{t,d}$ is the raw count of a term in a document, $\sum_{t' \in d} f_{t',d}$ is the total number of terms in document p.

The inverse document frequency is used to measure if a term is common or rare across all documents. It is defined by the following equation,

$$idf(t,D) = log \frac{N}{|d \in D : t \in d|} \tag{5}$$

where D is the entire collection of documents, N is the total number of documents in D, $\frac{N}{|d \in D : t \in d|}$ is the number of documents where the term t appears.

With the combination of tf value and idf value , we can define the tf-idf value. It is calculated by the following equation,

$$tf - idf(t,d,D) = tf(t,d) * idf(t,D) \tag{6}$$

In our project, we are confronted with a more specific problem. The web pages we get are now all related to covid, which means that our keyword(covid) is common across all web pages. Hence, it is not appropriate for use to compute IDF. In this case, we decide to use TF part of the TF-IDF algorithm as a measure of relevance.

The higher the tf value is, the more relevant the web page is to our keyword(covid).

3.3 The combination of the PageRank value and the tf value

In this project, we are playing as a search engine that is intended to find the most important and the most relevant web pages for our query(e.g. query is Covid-19). Hence, we plan to use the weighted sum of the PageRank value and the tf value as the metric used to rank the web pages. We call this score as PT score of the web page. It can be calculated by the following equation,

$$PT(p) = a * PR(p) + b * tf(p) \tag{7}$$

The values of a and b vary depending on the problem we need to solve. It should be determined by the users.

3.4 Page Summarizing

We had to optimize how we conducted the sentiment analysis on each page due to the volume of data, the restricted computing power, and the time required for the sentiment analysis model. The most effective approach to do this is to decrease the volume of text that the sentiment analysis model is input, which is what we intend to accomplish by summarizing each page. The summaries of the articles must precisely reflect the content of the articles due to the drastic decrease in text, making the selection of the relevant model for the summaries even more crucial.

The very recent Transformer model "T5 Transformer" is the recent state-of-the-art model in any kind of text-to-text sequence processing task. This transfer learning-based transformer model is largely used in many applications like Text generation, Text summarizing, Machine Translation, Chat-bots, and many more.[4]

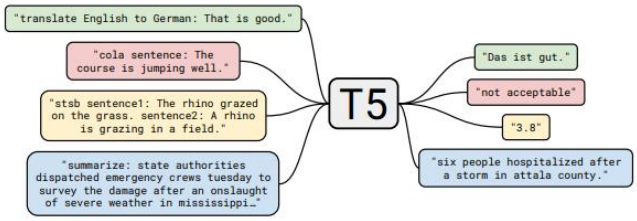

Figure 3: A diagram of T5 text-to-text framework[8]

T5 takes in data in the form of text only. And the outputs are also strings of characters, i.e. text again. This text-to-text nature enables the model to learn any NLP task without changing the hyperparameters and loss functions. Also it is "unified" in the sense that it can perform several NLG tasks simultaneously. It does not require separate output layers for different tasks like other transformers such as BERT and GPT2. Even for regression tasks, the output can be produced as a string of numbers. It has been trained upon a multi-task mixture of unsupervised and supervised tasks and for which each task is converted into a text-to-text format. For unsupervised training, it is trained on a novel 750 Gigabyte-huge dataset, the Colossal Clean Crawled Corpus(C4), which was created especially for T5. For supervised training, several popular datasets were used for respective tasks. Due to the above reasons, T5 model has out-performed several other contemporary SOTA architectures in different NLP tasks(at least in twenty different tasks) like text summarization, Question Answering, Sentiment analysis etc.[5]

3.5 Sentiment Analysis

For this project, there are three demands in sentiment analysis section: Need to process large amounts of data; Requires natural language processing; Requires a certain level of efficiency. Based on these three aspects, we found Spark NLP from John Snow Labs meet our requirements perfectly. It provides both Sentiment detection and entity recolonization with large scaled data, which can help us to get key words and public emotion in different time from processed data.

Then the processed data goes into bert_sequence_classifier _emotion Model, we use it because it was pre-trained using

a large dataset (Hugging Face), has a remarkable level of accuracy when applied to natural language processing tasks to analyze emotions, and only requires small tuning when applied to our project. For instance, it achieves a validation accuracy of 0.931 in the published documentation.

bert_sequence_classifier_emotion is based on the Bert model with an extra linear layer and activation function. BERT (Bidirectional Encoder Representations from Transformers), which enables researchers to obtain state-of-the-art performance on numerous NLP tasks by fine-tuning the representations on their data set and task, without the need for developing and training highly-specific architectures[6].

3.6 Keywords Extraction

We use YAKE! to extract keywords from each page, which is a light-weight unsupervised automatic keyword extraction method which rests on statistical text features extracted from single documents to select the most relevant keywords of a text. This system does not need to be trained on a particular set of documents, nor does it depend on dictionaries, external corpora, text size, language, or domain. To demonstrate the merits and significance of YAKE!, there is a compression between YAKE! and other ten state-of-the-art unsupervised approaches and one supervised method, as shown in Table 1.

Table 1: Comparison YAKE! and other algorithms in retrieving keywords[2]

Method	Precision	Recall	F1-Measure
YAKE-n3	0.029	0.104	0.046
YAKE-n4	0.010	0.138	0.018
YAKE-n5	0.006	0.148	0.012
TF-IDF	0.002	0.036	0.004
TextRank	0.001	0.079	0.003
TopicRank	0.009	0.032	0.014

Experimental results carried out on top of twenty datasets show that YAKE! significantly outperforms other unsupervised methods on texts of different sizes, languages, and domains.[2]

4. FINAL GOALS

In this section, we shortly describe what would our product be, and what the results look like.

4.1 Data Product

We wanted to implement a public-facing presentation page whose content contained a certain level of interactivity. By dragging the different points on the timeline, the user will be able to see what the hot topics, public attitudes and prevailing views on the new coronavirus are for that month.

4.2 Visualization

As this project required to demonstrate data product in static HTML website only. Therefore, we will try to use a highly integrated python data visualization API that can be embedded in HTML and used in Jupiter-notebook and Data-brick. For example, we intend to use Wordcloud, a python API which can generate word cloud with given context and kick it to HTML , to create word clouds for each period of flitted data. In addition, we will design a timeline on the web page with a number of points on the axis, each representing a specific month as Figure 4. When the user clicks on this point, the word cloud and the dominant voice for that month will pop up, and the content shown here is the result of filtering, processing and analysing the raw data.

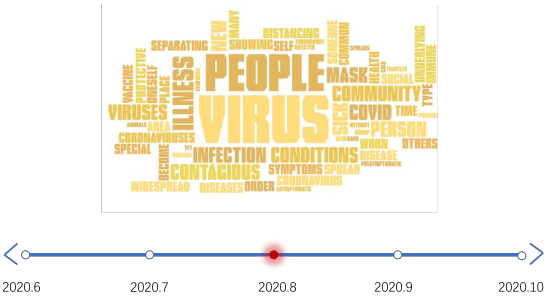

Figure 4: Words clousd in different time period

Another visual API that we intend to use comes from Pygal, which allows users to create beautiful interactive diagrams that can be converted into SVG format at the optimal resolution for printing or displaying on a web page using Flask or Django. There are many plots drawn with Pygal during the data exploration, and some of them are quite informative, for instance, in Figure 5 the quantity of URLs linked to Covid from the UK in some way illustrates the varying levels of public interest in the coronavirus.

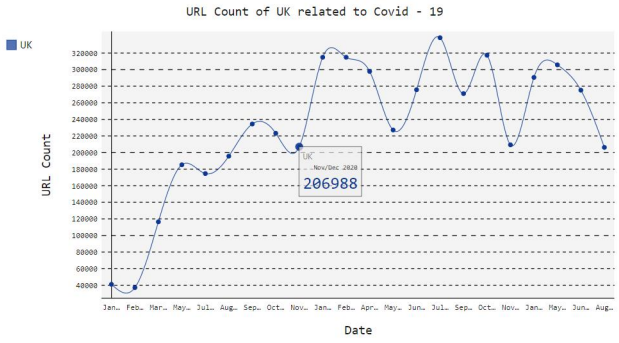

Figure 5: Number of URLs associated with Covid

For the visualisation of PageRank, the PageRank output contains citation relationships between pages and we can create topology maps based on this data. It looks like Figure 6.

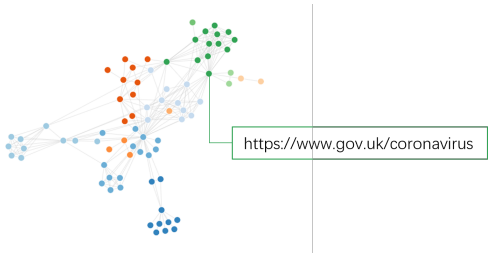

Figure 6: Visualization of PageRank

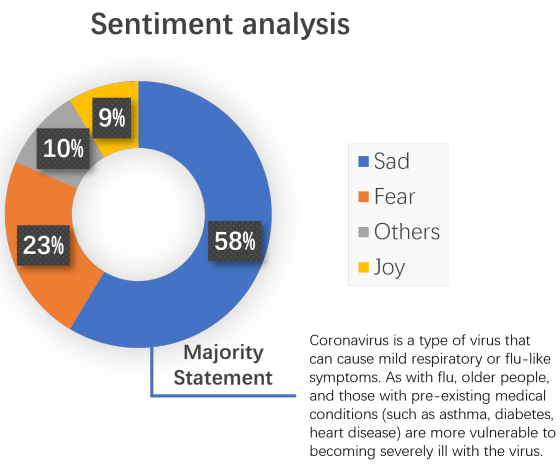

Figure 7: Visualization of Sentiment Analysis

From this structure, we can see the relationships between web pages. In addition, we plan to list the most important 10 web pages of one month based on the PT score(see section 3.3). With our data products, people can know how the pages about Covid-19 are connected to each other and also the most important pages at a given period. They can use this information to help them get a quick overview of Covid-19.

For the visualisation of Sentiment analysis, it may looks like Figure 7. In Figure 7 we can identify the sentiments expressed on each page and the words that go with them from the results of the sentiment analysis, and we want to visually represent the percentage of each sentiment and the key phrases associated with it for each month.

With our products, one can see how people's emotions about Covid-19 change over time and what specific views people have. It may help some people analyse the effect of Covid-19 more easily.

We intend to save the visualization results of both PageRank and sentiment analysis as SVG format, As they are images in an interactive form, they can easily be added to front-end pages without the need to write additional JavaScript.

5. ESTIMATED CALCULATION OF PROJECT COST

Before estimating our project's cost, we first look at how big our data is and how much computing time we need. Roughly, we need to access WAT and WET files related to the Covid-19 subject in the United Kingdom. Their size volume is about 41 GB(27.3 GB for WAT files, 13.7GB for WET files), and there are about $3,519,340$ web pages. We need to use them to measure the importance of each web page and do sentiment analysis. For WAT files, we need to extract links on every web page, build a graph whose nodes are web pages, and compute PageRank value for every node. After extracting links, we probably get a much smaller dataset. Let us assume its size is one-tenth of the original dataset, which is about $27.3 * 1/10 = 2.73$GB. For WET files, we only need plain text information which accounts for roughly 90% of the data, that is $13.7 * 0.9 = 12.33$GB. In a word, we have about $12.33 + 2.73 + 41 = 56.06$GB data needed to be stored in S3 buckets.

For sentiment analysis, we first need to preprocess the plain text data, and transform them into appropriate formats. After that, we apply a sentiment analysis algorithm to it. We roughly estimate the time required to perform sentiment analysis on a web page which is about 25 seconds. Currently, it is really hard to estimate the time of executing the queries since we are using resources of indeterminate size (dynamically allocated by the spark cluster). We assume we used roughly 2 GB of memory.

5.1 EC2 Cost

The price for an i3.large server is \$0.0558 per hour. Currently, we plan to choose four i3.large instances, and run about 315.5 hours, which would cost about $315.5.5 * 0.0558 * 4 \approx \$140$. Every i3.large instances have 15.25 GB memory and two CPU cores. Although it seems that using two i3.large instances is enough for our project, but for the sake of additional operations that may occur later, choosing 4 instances is a better choice.

5.2 S3 Cost

For S3 Standard plan, the price for the first 50TB stored for one month is \$0.0245 per GB. The price for PUT, COPY, POST, LIST requests (per 1,000 requests) is \$0.0054. The price for GET, SELECT, and all other requests (per 1,000 requests) is \$0.00043.

In our case, we currently need to store about 56.06 GB of data which would take about $\$56.06 * \$0.0245 = \$1.375$. In addition, we take the cost of requests into account. Roughly, we need to access all data stored in S3 buckets which means that we need to perform LIST and SELECT at least once for each WAT file and WET file. Hence, the price paid for requesting is about $(\$0.0054 + \$0.00043)/1000 * 3519340 \approx \$20.52$. The total of S3 cost is $\$20.52 + \$2.335 = \$22.855$.

5.3 Estimated Cost in Total

Summarizing the calculations above, we get a very rough estimation which is about $\$140 + \$22.855 = \$162.855$. It may meet the basic project requirements, however, it might be incorrect.

6. PROJECT TIMELINE

We also planned the way our project would move forward and what goals would be accomplished in what time frame. The project timeline is shown in Table 2,

Table 2: Project Schedule

Date	What we need to do
10.4-10.10	Data pre-processing, implement all algorithms we need in a small dataset
10.11-10.17	Improve our pipeline, and then apply it to the entire datasets
10.18-10.24	Build the static HTML website, complete data visulazation
10.24-10.30	Write slides, do the presentation, write the report, and improve the code and the HTML website

7. REFERENCES

[1] Warcio: Warc (and arc) streaming library. https://github.com/webrecorder/warcio.

[2] R. Campos, V. Mangaravite, A. Pasquali, A. Jorge, C. Nunes, and A. Jatowt. Yake! keyword extraction from single documents using multiple local features. *Information Sciences*, 509:257–289, 2020.

[3] A. Dave, A. Jindal, L. E. Li, R. Xin, J. Gonzalez, and M. Zaharia. Graphframes: an integrated api for mixing graph and relational queries. In *Proceedings of the fourth international workshop on graph data management experiences and systems*, pages 1–8, 2016.

[4] I. Ganguli, R. S. Bhowmick, S. Biswas, and J. Sil. Empirical auto-evaluation of python code for performance analysis of transformer network using t5 architecture. In *2021 8th International Conference on Smart Computing and Communications (ICSCC)*, pages 75–79, 2021.

[5] K. Grover, K. Kaur, K. Tiwari, Rupali, and P. Kumar. Deep learning based question generation using t5 transformer. In D. Garg, K. Wong, J. Sarangapani, and S. K. Gupta, editors, *Advanced Computing*, pages 243–255, Singapore, 2021. Springer Singapore.

[6] D. Nozza, F. Bianchi, and D. Hovy. What the [mask]? making sense of language-specific bert models. 03 2020.

[7] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[8] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

[9] A. Rajaraman and J. D. Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.

[10] R. S. Xin, J. E. Gonzalez, M. J. Franklin, and I. Stoica. Graphx: A resilient distributed graph system on spark. In *First international workshop on graph data management experiences and systems*, pages 1–6, 2013.