Text Preprocessing and its Implications in a Digital Humanities Project

Maria Kunilovskaya
University of Wolverhampton, UK
University of Tyumen, Russia
maria.kunilovskaya@wlv.ac.uk

Alistair Plum
RGCL
University of Wolverhampton, UK
a.j.plum@wlv.ac.uk

Abstract

This paper focuses on data cleaning as part of a preprocessing procedure applied to text data retrieved from the web. Although the importance of this early stage in a project using NLP methods is often highlighted by researchers, the details, general principles and techniques are usually left out due to consideration of space. At best, they are dismissed with a comment "The usual data cleaning and preprocessing procedures were applied". More coverage is usually given to automatic text annotation such as lemmatisation, part-of-speech tagging and parsing, which is often included in preprocessing. In the literature, the term 'preprocessing' is used to refer to a wide range of procedures, from filtering and cleaning to data transformation such as stemming and numeric representation, which might create confusion. We argue that text preprocessing might skew original data distribution with regard to the metadata, such as types, locations and times of registered datapoints. In this paper we describe a systematic approach to cleaning text data mined by a data-providing company for a Digital Humanities (DH) project focused on cultural analytics. We reveal the types and amount of noise in the data coming from various web sources and estimate the changes in the size of the data associated with preprocessing. We also compare the results of a text classification experiment run on the raw and preprocessed data. We hope that our experience and approaches will help the DH community to diagnose the quality of textual data collected from the web and prepare it for further natural language processing.

1 Introduction

In this paper we discuss our experience of preprocessing textual descriptions of cultural events collected from a number of diverse web sources for a DH project. The problems that we faced demonstrate that there is a big difference between the amount of raw data scraped from the web and the amount of data distilled from it through preprocessing for further application of natural language processing (NLP) methods. The web is a unique source of available information that, to an extent, reflects social and cultural trends, and it is increasingly used in sociological and cultural studies. However, it seems that preprocessing and filtering, necessary to ensure reliability of automatic analyses, creates an additional refraction of the original signal registered by the web. This paper offers a description of the data preprocessing stage in a DH project and reports its effects on the quality of text classification on the one hand, and on the size of data available for the analysis as compared to the original web-scraping results on the other hand. We used pre-trained ELMO (Embeddings from Language Model) models for text representation in the classification task. We aim to find a reasonable trade-off between quantity and quality, and try to salvage as many text instances as possible without compromising the performance of NLP methods.

Our specific tasks include:

1. description of a preprocessing pipeline and its application to the data at hand;

2. evaluation of the impact of preprocessing on text classification results;

3. estimation of the impact of preprocessing on the dataset integrity.

The data underlying this project includes texts in at least five languages, with a lot of cases of code-switching. However, for the purposes of the paper we focus on English and Russian subsets of the data.

2 Related work

The term 'data preprocessing' is used as an umbrella term to refer to a wide range of procedures, including cleaning, filtering, normalising, annotating and transforming raw data. It is not always clear which operations from this list are referred to in a particular description. For example, functions included in a *Keras* preprocessing module convert files in a tree of folders into the *Keras* internal dataset format by creating batches of texts and linking texts to labels, and possibly truncating text to a pre-set maximum length. It assumes that the text files in folders are already suitably prepared for machine learning experiments. In this paper, we focus on the steps that precede this sort of data transformation and have to do with ensuring structural completeness of the data, its cleaning and normalisation.

The link between the quality of the data and the reliability of the research outcomes is well known. It is reflected in the popular 'garbage in, garbage out' wisdom shared by the research communities in data science and NLP. It was famously formulated by John Sinclair, a widely-recognised authority in corpus building and corpus linguistics: "the beginning of any corpus study is the creation of the corpus itself ... The results are only as good as the corpus" (Sinclair, 1991).

Data collection and preprocessing is not only an important stage in a research project, it is also the most tedious and time-consuming step. A popular belief is that it claims up to 80% of the research time. According to a report by *Anaconda* "2020 State of Data Science: Moving From Hype Toward Maturity", based on a survey, which attracted a total of 2,360 responses from researchers from over 100 countries says that "on average 45% of their time is spent getting data ready (loading and cleansing)". [1] Based on the previous research, there is no getting away from this chore, especially if using bag-of-words representations.

Preprocessing is always dependent on the data and task at hand, and it is often so specific to a project that it is not described in detail. Nevertheless, there have been numerous attempts at evaluating the effects of different preprocessing steps for different types of data and NLP tasks. Subramaniam et al. (2009) present a survey of different sources of noise in texts and how these types of noise can be handled. The survey cites techniques such as automatic speech recognition, optical character recognition and machine translation to be the biggest culprits in terms of introducing noise. Kathiravan and Haridoss (2018) go into detail about preprocessing techniques used specifically for information extraction (IE) and information retrieval (IR). While this study utilises all the techniques mentioned previously, it focuses on different stemming approaches, as it is said to be one of the most beneficial preprocessing steps in terms of performance.

In Sergienko et al. (2016) 'text preprocessing' refers to data handling operations such as stop-words filtering with stemming, feature selection, feature transformation and a novel transformation method based on classes of terms. The study proves that using the novel feature transformation method and ensembles of term weighting methods yields good classification results even with very small amounts of features. Similarly, Gonçalves and Quaresma (2005) reports the impact of non-relevant word removal and lemmatisation/stemming in conjunction with data transformation steps in the context of text classification for two languages. Using a Support Vector Machine (SVM) classifier to represent Portuguese data from the law domain as well as English Reuters texts, experiments were carried out on feature reduction and construction, feature subset selection and term weighting. In terms of feature reduction, three configurations were tested including different combinations of non-relevant word removal (including articles, pronouns, adverbs and prepositions), as well as lemmatisation and stemming. Crucially, the authors found that for Portuguese stop-word removal did yield better results, while for English stemming had to also be included to attain better results. Kadhim (2018) propose similar methods for tf-idf based text classification, but also include the removal of HTML tags.

More recently, HaCohen-Kerner et al. (2020) performed an extensive evaluation of common preprocessing steps before text classification. Exploring the impact of all possible combinations of five to six basic preprocessing methods on four benchmark text corpora, the study tested three machine learning methods. The preprocessing methods included spelling correction, conversion to lower case, HTML tag removal, punctuation removal, stop-words removal and reduction of replicated characters. Overall, it is determined that the best

[1] https://www.anaconda.com/state-of-data-science-2020

results could be achieved with a combination of preprocessing methods, while stop word removal to be the most effective preprocessing method that significantly improved text classification. However, the exact combination of methods is said to vary significantly across datasets, leading to the conclusion that every dataset has to be treated accordingly.

The impact of noise on the outcomes of NLP tasks, implemented using BERT-based deep learning methods, such as offensive language identification, informativeness identification and sentiment analysis, is explored in Al Sharou et al. (2021). In line with the above studies, they limit the concept of noise to non-standard appearance of language units (casing, hashtags, code-switching, emoji, URL and punctuation). They found that for some tasks these types of noise are actually useful, and filtering them out results in lower classification results.

Most of the related work is focused on handling text representations such as feature reduction and selection, with some emphases on stop-words removal and lemmatisation/stemming. However, most researchers worked with available datasets that have undergone some sort of preprocessing. In our setting, we dealt the immediate results of crawling and had to extract raw text from it.

3 Data origins and structure

The textual data underlying this study is part of a dataset contracted by a research group for a DH project and collected by a data company following a technical assignment. The dataset includes tables, where each row represents a cultural event announced at one of the seven web services aggregating news about cultural life in various locations across the globe. The project is set within comparative cultural studies and focuses on the types of events advertised in Russia and elsewhere in the world. That is why the sources of data include both international and national publishing platforms, social networks and distributors of information regarding cultural events. A cultural event is understood broadly as any social (public) activity in the spheres of contemporary arts, education, sports, music, theatre, including social activities arranged by restaurants, computer clubs, and private enthusiasts (yoga classes, guided tours, etc). There is an understandable imbalance between the number of cultural events coming from international and Russian sources (see Table 1.

Table 1 has the names and links to these resources as well as the number of raw data points acquired from each website. The content of the web pages was either scraped (e.g. e-flux, Theories and Practice), or accessed through the official API (e.g. Behance, Timepad), or downloaded using open data service (Russian Ministry of Culture).

Data collection did not target any particular time period, but acquired all available announcements. Most of the platform's archives date back to 2003-2004. The data was collected up to mid-2019.

The records of events obtained from these web resources included city and date of the event and its textual description. Besides, each event was characterised by the name of the website section, where it was published, if available (e.g. 'business', 'graphic design', 'lectures'). This raw data was supplemented with additional information about event locations. The diverging thematic groups of events from the websites were manually mapped into seven major categories. The enhanced data tables have 24 columns, which store objective properties of each location: unique event identifier, date, city, synthetic and raw semantic categories, geographic coordinates, country, region, population, social and economic development parameters of the area (GDP, human development index, economic complexity index).

4 Preprocessing Methodology

4.1 Structural Completeness

The first obvious step to take is to ensure the overall structural sanity of the dataset. It makes sense to see whether there are empty values in the columns that are of primary importance for the subsequent analysis. Our project is centered around analysis of descriptions in the 'text' column in conjunction with 'date' and 'city'. Working in *pandas*, one of the most versatile Python libraries for tabled data, we collected the counts of NaN values in each column *i*: (`counts = df[i].isnull().sum()`) into a dictionary keyed on the column names and determined the ratio of NaN per column. It turned out that the text column had 1.67% of empty values (75,918 datapoints). These need to be dropped to avoid further errors in processing.

Similarly, the data stored in text column can be all numbers and contain no alphabetic characters. In this case, this content is hardly of value for subsequent semantic analysis. We found the indices of rows with this issue in 'text' col-

Source	Description	Region	Data points
Behance `api.behance.net/v2/users/X/projects`	Adobe's social media, advertising, architecture, fashion, industrial design	international user profiles	1,184,509
e-flux `www.e-flux.com/announcements`	a publishing platform and archive, artist projects focused on visual arts, inc. education	North America, Europe	10,280
Meetup `www.meetup.com/api/general`	a social media platform for building local communities and discovering what is happening nearby	international, user information	2,635,724
Ministry of Culture `opendata.mkrf.ru/opendata/7705851331-events`	a monitoring archive of cultural events registered by companies and individuals	Russia	84,222
TED `www.ted.com/tedx/events`	announcements of TED-style local events	international	26,001
Timepad `afisha.timepad.ru/`	a online service which helps to manage, promote and attend events	Russia	524,432
Theories and Practice `theoryandpractice.ru/seminars`	announcements for seminars, courses, workshops, master classes	Russia	70,379
Total			4,535,547

Table 1: Sources of cultural data and their coverage

umn with idx=df['text'].str.contains(pat='[a-zA-ZA-Яа-я]', regex=True) and sliced the dataframe on this index to lose additional 18,243 instances (0.4% of the original data size). This also filters out texts entirely in scripts other than Latin or Cyrillic.

Finally, in most cases automatically collected web data is known to carry duplicates, which can unfairly inflate counts of some items. One way to retain only the unique entries is to run `duplicated = df[~df.duplicated(keep='first')]`. A more rigorous approach is to drop rows that have duplicates in the focused column ('text' in our case): `texted.drop_duplicates('text', inplace=True)`.

All preprocessing decisions need to be taken with the view of the main research tasks. For example, it can be important to retain duplicate announcements that appeared on several platforms or on several dates. They indicate re-current events or events with enhanced visibility.

These operations can be computationally demanding, with less powerful computers getting stuck, if all data is fed to one CPU in one processing thread. It makes more sense to partition big datasets and process them in parallel. Our data was naturally split by source, i.e. by the web resource of data origin. Keeping these datasets apart instead of collecting them into one massive file was useful to understand the quality of the data coming from various sources. The overall size of our data was 4.6 GiB uncompressed, with one of the files accounting for 3.8 GiB and throwing out-of-memory error. We split it into 5 files for easier processing. Besides, we found it helpful to perform each preprocessing step as a separate process with its own intermediary output. It gives better control over the preprocessing workflow and allows to fall back to the output of the previous step as a contingency measure in case of errors, instead of running the whole preprocessing pipeline from scratch. For example, we found that some preprocessing libraries require switching to earlier versions of Python than our standard setup. At the time of writing, 'contraction' is available for up to Python3.6 and ‘unidecode’ up to Python3.7. Additionally, to save disk space we suggest working with single-member archives for each input file. In pandas, it is loaded and saved by adding the keyworded argument (compression=‘gzip’) to the standard commands: `df = pd.read_csv(data1.tsv.gz, compression=‘gzip’)` and `df.to_csv(‘data1.tsv.gz’, compression=‘gzip’)`.

After this initial clean-up step, which is a necessary minimum preprocessing required regardless of the input language, our data shrank by 19% of the original data size.

4.2 Cleaning and Noise Reduction

Language modeling and analysis by NLP techniques require some type of numeric representation of texts. This includes indexing, one-hot representations, frequency-based approaches (e.g. tf-idf), vectorisation based on co-occurrence statistics, which exploits the distributional hypothesis in lexical semantics. We assume that some NLP methods work better when the input text is free of extraneous artefacts, such as HTML tags, encoding errors, literal renditions of Unicode characters such as `\xe2\x80\x93` and `\r\n\t`. They appear in the top of a sorted frequency dictionary built from the raw text and may affect the quality of lemmatisation, which was shown to be useful for automatic semantic analysis in previous work. Although URL and email addresses can be useful for some tasks, we treated them as organisational information, external to the text, and removed them using regular expressions.

The sequence of the operations aimed at removing or reducing noise can be important both for safety and efficiency of the process. For example, if you are prepared to drop texts, shorter than five tokens, it makes sense to run this command for the first time early on in the stack of preprocessing operations to exclude these instances from further processing. Each web resource selected for scraping together with the scraping settings is likely to produce individual types of noise that can be detected by looking at the top of the reverse-ordered frequency dictionaries generated after each procedure or preprocessing step. As a result of several attempts and recursive iterations, we developed a preprocessing pipeline, which includes the following operations:

- insert a space (i) before a capital, following a lowercase character and (ii) after a set of punctuation marks (both are usually a sign of a missing delimiter, such as a newline, e.g. What we’ll doGoogle Cloud Study Jams What to bringLaptop);
- drop recurrent commenting phrases typical for some websites (e.g. ‘Meeting description’ and square-bracketed content (e.g. [masked]);
- drop unnaturally long tokens (over 15 characters), usually a sign of malformed content;
- normalise double and single quotes, dashes, hyphens, tildes, dots, apostrophes;
- remove multiple spaces and all characters, except an allowed set to filter out incorrectly decoded Unicode (e.g. ÄéÊÉÖÂ±ÂÖ) (using a function adapted from the processing pipeline developed by Shavrina and Shapovalova (2017);
- remove tokens that contain no alphanumeric characters, usually noise and orphaned punctuation marks;
- fix literal renditions of HTML entities such as `\&gt`, `\&apos;`;
- fix spaced or mostly spaced strings (e.g. E S H K O L O T F E S T I V A L);
- remove unmotivated repetitions of characters and words;
- weed out items with unmotivated high frequencies unique for each web resource.

It makes sense to finish this stage of language-independent preprocessing with running a language detection module and storing its output in a new attribute (column) to be used at the lemmatisation stage. One major problem with language detection is that some texts in our dataset include mixed code, i.e. they have words written in several languages (e.g. ‘Илья Чёрт в The Right Place’, ‘New Year Mylene Farmer Fan-Club Party в ночном клубе “Jack Jan”). This can be an issue for any automatic language detection. We used an additional heuristic (checking whether any characters are Cyrillic) to correct the prediction of the automatic language detection for mixed code samples, produced by *fasttext* language detection model [2]. Following language identification we dropped all observations in languages other than English and Russian. The number of texts in other languages (Italian, French, Spanish) was disproportional to the bulk of the data, and cumulatively accounted for about 7%.

4.3 Text Normalisation

Machine learning algorithms and lemmatisation pipelines perform better, if the textual data is standardised, e.g. spelling variants are replaced with a one unified convention. Typically, this is done for quotation symbol styles, English contracted forms, and Russian ë. When choosing a specific type of normalisation, it is good to have in mind the requirements to the input at the next stage of data processing, particularly, which linguistic annotation tool (e.g. for lemmatisation) and which numeric representation method is planned for the main study. For example, the Russian model learnt on SynTagRus treebank (Droganova et al., 2018) does not recognise «», one of the double quotes styles, used in Russian texts. We have taken care of quotes at the previous step. In this project, language-specific normalisation concerned expanding English contractions (admittedly an optional task, given the subsequent use of UDpipe, which can process contractions) and normalisation of Russian ë. In cases when lemmatisation does not follow, normalisation measures can include bringing all words to lower case, and stop-words and punctuation removal. Finally, at this stage we filtered out all datapoints with the textual attribute shorter that 5 tokens.

4.4 Automatic Lemmatisation and Filtering

For lemmatisation, we relied on Universal Dependencies (UD) framework using UDPipe module (v1, Straka and Straková, 2017), with the Russian model trained on SynTagRus treebank (v2.5, Droganova et al., 2018). All tokens in our texts were replaced with their lemmas identified by the parsing pipeline. UD-annotations were also used to produce and store lists of lemmas of content words (tagged as NOUN, ADJ or VERB) from each text to be used in solving the main research task, i.e. grouping events by category in an unsupervised machine learning setting. Content lemmas were filtered with stop-words lists [3]. At the time of annotation we discarded instances that had no content words.

5 Results and Discussion

To demonstrate the impact of preprocessing and annotation on the outcomes of the experiments, we compared the results of automatic classification on the texts at different stages of preprocessing (raw tokens, clean tokens, all lemmas, content lemmas). To this end, we selected a subset of 600 event descriptions (in Russian) registered in Tyumen (a big city in Western Siberia, Russia). These datapoints were labelled with the names of the website sections where the respective event was announced (*Timepad* and *Ministry of Culture*). The six categories used as class labels were 'theatre', 'psychology and self-development', 'business', 'sport', 'concerts', 'kids events' and had at least 30 observations in each category. The texts are represented using available ELMO models (Kutuzov and Kuzmenko, 2017), pre-trained on raw and lemmatised corpora matching the types of preprocessing we have in our data. Each text received an averaged vector for all words in it. The ELMO vectors (size 1024) were then transformed into 32-dimensional vectors with *Autoencoder* module from the *Keras* library for Python. These representations were classified with a SVM algorithm with the default *scikit-learn* parameters. The evaluation is performed in 5-fold cross-validation setting. Given the imbalance in our data, we used stratified folds and weighting for underrepresented classes. Table 2 reports accuracy and macro F1-score for each of the preprocessing variants.

	accuracy	**F-score**
raw input	76%	0.72
cleaned tokens	76%	0.72
lemmatised text	80%	0.76
content lemmas	82%	0.78

Table 2: Classification results on versions of input

As can be seen from Table 2, there is no difference between the classifier performance on raw and cleaned tokens, but lemmatisation definitely helps. It is not clear whether preprocessing procedures like those described in Sections 4.2 and 4.3 improve the quality of lemmatisation, but it seems that vectorisation effectively coped with raw text. Frustratingly, all the tedious cleaning procedures did not pay off. One explanation for this counter-

[2] https://fasttext.cc/docs/en/language-identification.html

[3] https://github.com/Alir3z4/python-stop-words

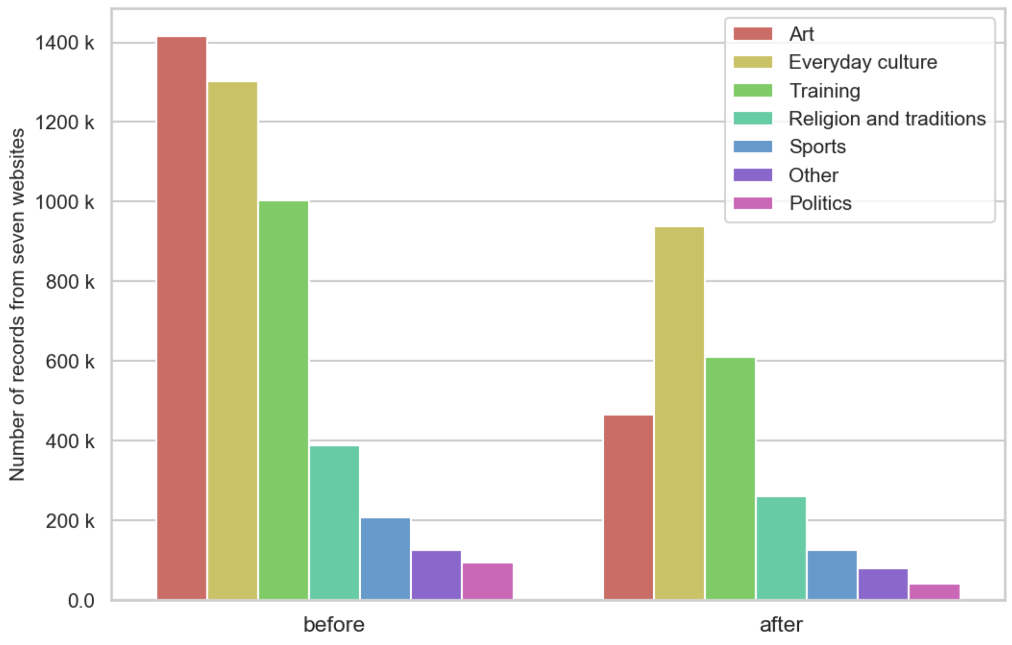

Figure 1: Distributions of events across categories before and after preprocessing

intuitive result is that the Russian data in part was downloaded from an official government-curated archive. The quality of these texts was higher in comparison to the typical user-generated content seen on other web resources.

Another alarming aspect of preprocessing is the portion of the original dataset that gets discarded during preprocessing. Overall, for the dataset at hand, preprocessing claimed 45% of the original size in terms of the number of observations. Importantly, preprocessing affected the ratio of events assigned to various categories. The diverging inventories of categories used for sectioning on the websites were manually mapped into seven thematic groups shown in Figure 1. Figure 1 demonstrates the impact of preprocessing on the distribution of records across the seven thematic categories from all seven web resources. It can be seen that preprocessing resulted in a significant reduction in the 'art' and 'politics' categories (62.7 and 57.4% of registered events lost, respectively). Most of the items lost in the 'arts' category had a link to a media object such as a picture in the 'text' field. It raises a question whether any multi-modal processing can be employed as a contingency measure.

The top three web resources that lost most of the records in preprocessing were *Behance*, *Timepad* and *TED*, which were reduced by 81.9, 54.1 and 55.3% respectively. The highest integrity was characteristic for *e-flux* and *Ministry of Culture*; they had little noise and hardly lost any datapoints. Note that in terms of absolute number of datapoints, *Meetup* remained the biggest source of data, dwarfing *e-flux* into insignificance (with only 10,280 raw records and 10,081 after pre-processing, it is shown as a dot in Figure 2.

For the Tyumen sample, we are left with less than 50% (992 events) of the original number of observations (1864 events), given the preprocessing described above. The number of observations for Manchester (for example) dropped by 35%. Most of the datapoints are discarded due to the lack or insufficiency of text attributes associated with recorded events. Interestingly, in another experiment we set the text lengths threshold to 10 words (as opposed to 5 here) and achieved the classification accuracy of over 90% on just 452 observations, represented as content lemmas, in the same experimental setup. On the one hand, it seems to indicate that a smaller, but higher-quality data returns better results, but on the other hand, can we claim that the results pertain to the original dataset? If half of the datapoints end up in the garbage bin, how can we say that the results reflect the content of the web

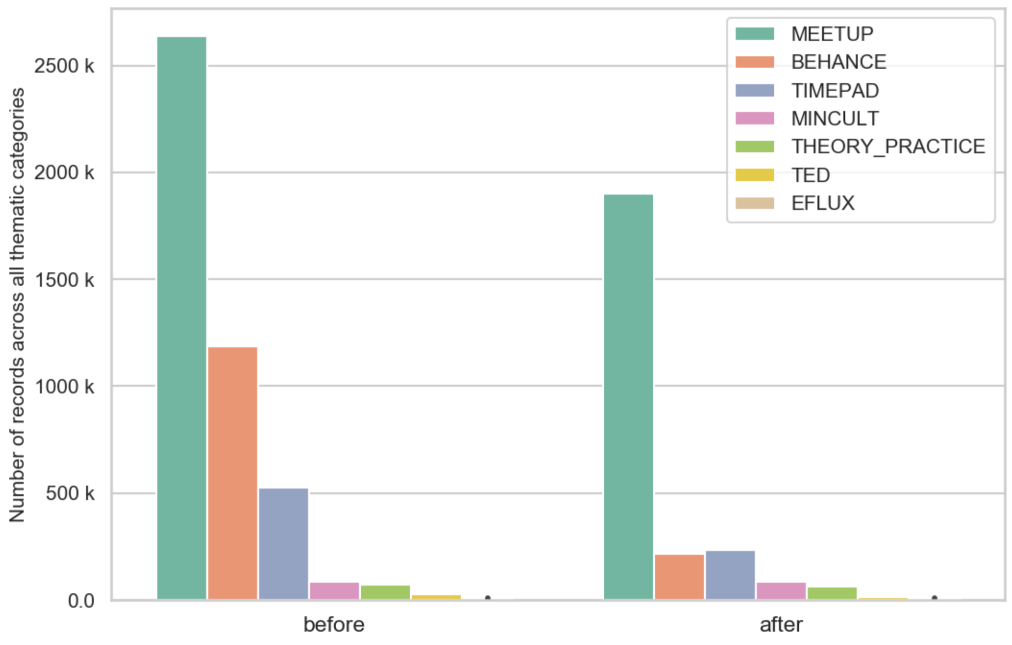

Figure 2: Number of records from seven web resources before and after preprocessing

sources used? Our observations raise issues with the selection of web resources and data collection techniques. We have seen that some websites are more suitable as sources of textual data than others.

6 Conclusion

The paper contains a detailed description of preprocessing steps that were taken to prepare textual data collected from seven web aggregators of cultural information for automatic analysis using NLP methods. We followed the best practices in the field, and expected that cleaner data with 15-40% smaller vocabulary would yield gains in the classifier performance proportional to the efforts invested. However, it seems that these efforts fall with the 80% of work that generates only 20% of profits, following the famous Pareto principle. In the future, we plan to explore the interrelations between types of preprocessing and actual gains seen for different types of language representations, particularly embeddings and classification setups. What is the impact of preprocessing on parsing? Which types of noise can actually be tolerated? One take-away from this project is that selection of web resources is important: some of them return cleaner data than others. It is true that data preprocessing takes most of the research time, too. When reporting the results, in addition to the inaccuracies of analysis, two caveats have to be made with regard to the amount of observations (datapoints) discarded during preprocessing.

Acknowledgements

This work has been partly supported by the Russian Foundation for Basic Research within Project *Cultural Trends in the Tyumen Region in the National and Global Contexts* No. 20-411-720010 p_a_Tyumen region.

References

Khetam Al Sharou, Zhenhao Li, and Lucia Specia. 2021. Towards a Better Understanding of Noise in Natural Language Processing. In *RANLP-2021*, page in print.

Kira Droganova, Olga Lyashevskaya, and Daniel Zeman. 2018. Data conversion and consistency of monolingual corpora: Russian ud treebanks. In *Proceedings of the 17th international workshop on treebanks and linguistic theories (tlt 2018)*, volume 155, pages 53–66.

Teresa Gonçalves and Paulo Quaresma. 2005. Evaluating preprocessing techniques in a text classification problem. *São Leopoldo, RS, Brasil: SBC-Sociedade Brasileira de Computação*.

Yaakov HaCohen-Kerner, Daniel Miller, and Yair Yigal. 2020. The influence of preprocessing on text classification using a bag-of-words representation. *PloS one*, 15(5):e0232525.

Ammar Ismael Kadhim. 2018. An evaluation of preprocessing techniques for text classification. *International Journal of Computer Science and Information Security (IJCSIS)*, 16(6):22–32.

Periasamy Kathiravan and N Haridoss. 2018. Preprocessing for mining the textual data: A review. *vol*, 7:5–8.

Andrey Kutuzov and Elizaveta Kuzmenko. 2017. *WebVectors: A Toolkit for Building Web Interfaces for Vector Semantic Models*, pages 155–161. Springer International Publishing, Cham.

Roman Sergienko, Muhammad Shan, and Wolfgang Minker. 2016. A comparative study of text preprocessing approaches for topic detection of user utterances. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1826–1831.

Tatiana Shavrina and Olga Shapovalova. 2017. To the methodology of corpus construction for machine learning:«taiga» syntax tree corpus and parser. In *Proceedings of the "Corpora*, pages 78–84.

John Sinclair. 1991. *Corpus, Concordance, Collocation*. Oxford University Press.

Milan Straka and Jana Straková. 2017. Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99.

Venkata Subramaniam, Shourya Roy, Tanveer Faruquie, and Sumit Negi. 2009. A survey of types of text noise and techniques to handle noisy text. In *Proceedings of The Third Workshop on Analytics for Noisy Unstructured Text Data*, pages 115–122.