Twitter-Demographer: A Flow-based Tool to Enrich Twitter Data

Federico Bianchi
Stanford University
Stanford, California, USA
fede@stanford.edu

Vincenzo Cutrona
SUPSI
Lugano, Switzerland
vincenzo.cutrona@supsi.ch

Dirk Hovy
Bocconi University
Milano, Italy
dirk.hovy@unibocconi.it

Abstract

Twitter data have become essential to Natural Language Processing (NLP) and social science research, driving various scientific discoveries in recent years. However, the textual data alone are often not enough to conduct studies: especially, social scientists need more variables to perform their analysis and control for various factors. How we augment this information, such as users' location, age, or tweet sentiment, has ramifications for anonymity and reproducibility, and requires dedicated effort. This paper describes Twitter-Demographer, a simple, flow-based tool to enrich Twitter data with additional information about tweets and users. Twitter-Demographer is aimed at NLP practitioners, psycho-linguists, and (computational) social scientists who want to enrich their datasets with aggregated information, facilitating reproducibility, and providing algorithmic privacy-by-design measures for pseudo-anonymity. We discuss our design choices, inspired by the flow-based programming paradigm, to use black-box components that can easily be chained together and extended. We also analyze the ethical issues related to the use of this tool, and the built-in measures to facilitate pseudo-anonymity.

1 Introduction

Twitter data are at the heart of NLP and social science research (Steinert-Threlkeld, 2018), used to study policy and decision-making, and understand public opinion's consequences better. Its accessibility and the variety and abundance of the data make Twitter one of the most fruitful sources to experiment with new NLP methods, and to generate insights into societal behavior (Munger, 2017). Given that 199 million people communicate on Twitter daily,[1] it becomes fundamental to find ways to interpret this information better.

However, researchers often need more than pure text data to control for the effects of various covariates, stratify the data into sensible subgroups, and assess their reliability. Social sciences typically require a recourse to external variables like age or location to control for confounds. In addition, NLP research has shown that integrating socio-demographic information can improve a wide range of classification tasks (Volkova et al., 2013; Hovy, 2015; Lynn et al., 2017; Li et al., 2018; Hovy and Yang, 2021). By default, this information is not available, and a wide range of NLP tools have been developed to infer measures from the text (i.e., sentiment, syntactic structure: (Balahur, 2013; Kong et al., 2014, inter alia) and user (age, gender, income, person or company: (Preoţiuc-Pietro et al., 2015; Wang et al., 2019, inter alia).

Here, we introduce Twitter-Demographer, a tool that provides a simple and extensible interface for NLP and social science researchers. Starting from tweet ids (the common way to share Twitter data), the tool hydrates the original text and can enrich it with additional information like the sentiment of the tweets, topics, or estimated demographic information of the author, using existing tools. Twitter-Demographer builds on previous research (Wang et al., 2019; Barbieri et al., 2020; Wolf et al., 2020), but puts all these efforts together in one simple tool that can be used with little effort. Twitter-Demographer can be applied to extract information from different languages, as its default components are either multi-lingual or language-independent.[2] Twitter-Demographer has a simple API that can be used to add user-defined components quickly and effectively.

[1] https://s22.q4cdn.com/826641620/files/doc_financials/2021/q1/Q1'21-Shareholder-Letter.pdf

[2] Note, however, that the use of language-specific classifiers might restrict the usage to specific languages.

One of our goals is to provide and enforce the generation of **reproducible data enrichment pipelines** (i.e., they can be shared and produce the same results if components are kept the same). With data enrichment we mean the process of extending a dataset, e.g., adding new inferred properties, or disambiguating its content (Cutrona et al., 2019). Our flow-based infrastructure makes it easy to produce and share pipelines with other researchers to reconstruct the extended datasets.

Most importantly, inferring user-related attributes poses a privacy issue, even for research purposes. We implement several algorithmic **privacy-by-design** solutions to facilitate **pseudo-anonymity** of the users, and to reduce the chance that their personal data or identifiers can be used to identify natural persons.

We believe that Twitter-Demographer can help (computational) social scientists wanting to analyze the properties of their datasets in more depth, and provide NLP practitioners with a unified way to enrich and share data.

Contributions We introduce Twitter-Demographer, a new tool to enrich datasets of tweets with additional information. The extensible tool enables NLP practitioners and computational social scientists to quickly adapt their own datasets with the features required for a specific analysis. Twitter-Demographer encodes the resulting enrichment pipeline in a stable, shareable, and reproducible format and implements privacy-by-design principles.

2 The flow-based paradigm

The flow-based paradigm is a programming paradigm that uses a *data processing factory* metaphor for designing applications as networks of black-box processes (Morrison, 2010). The paradigm is helpful for data handling because it allows users to easily combine different black-box components in many ways, fitting different requirements time by time. Each component implements a specific task, takes some inputs, and returns some outputs. Many solutions employ this kind of paradigm (e.g., Apache NiFi[3]). These solutions are directed at experts like data engineers because they require some knowledge about the low-level details (e.g., how to handle data sources, data streams, and event-based executions).

The advantage of the flow-based paradigm is that users do not have to know the intrinsic logic of each block (hence black-box). They only have to focus on combining blocks to ensure the proper mapping between inputs/outputs of consecutive blocks. Indeed, the main disadvantages of manually building these pipelines are that (i) they require massive effort to be defined; (ii) they are sensitive to various hurdles, e.g., what happens if we cannot find one tweet or its location is unavailable? (iii) they are error-prone, with minor errors possibly tearing down entire pipelines, e.g., what happens if a Web service changes its exchange data format, or is no longer available?

Twitter-Demographer has been imagined as a low coupled set of components that operates on a dataset in tabular format (e.g., a Pandas DataFrame). Each component takes the dataset as input, applies some operations on it (e.g., adding columns), and returns the modified dataset. Components can be integrated into pipelines: we aim for high cohesion and low coupling principles to reduce possible errors at the component level. Each component exposes a set of required inputs (i.e., columns that must be contained in the input dataset) and a set of generated outputs (i.e., names of the new columns added to the dataset). Using this information, we can chain different components together to introduce dependencies (e.g., to run the sentiment analysis classifier, we need first to query Twitter and create a new column containing the text of tweets). Exposing the input and the outputs allows for the consistency between different components to be checked beforehand to avoid compatibility issues.

The flow-based setup makes it possible to replace any component with another one implementing the same task with a different logic, as long as the new component respects the communication interface (i.e., expected inputs and generated outputs). It is worth noting that the paradigm does not force a specific absolute order between components: a component requiring some columns as input (e.g., col_X, col_Y) must be placed in any position after the components generating such columns.

The goal of Twitter-Demographer is two-fold: 1) providing an easy-to-use interface for data enrichment and 2) providing a system that allows more expert users to re-use and modify existing components that are already implemented in Twitter-Demographer easily.

[3]`https://nifi.apache.org/`

```python
# create the demographer object
demo = Demographer()

re = Rehydrate(token)
me = NominatimDecoder()
st = SentimentClassifier(model_name)

# add the components
demo.add_component(re)
demo.add_component(me)
demo.add_component(st)

# run the pipeline
new_data = demo.infer(data)
```

Listing 1: Example of Twitter-Demographer basic usage. ‘data‘ variable is a simple DataFrame with one column containing the tweet ids.

While it is true that Twitter-Demographer is mainly based on the composition of existing tools, these are wrapped into off-the-shelf components that simplify the usage of the proposed analytics methodologies.

3 Twitter-Demographer

We show the class diagram of Twitter-Demographer in Figure 1. While Listing 1 shows an example application of the tool. Line 2 instantiates the Demographer object, which is responsible for handling the entire pipeline (i.e., it also performs compatibility checks on components). Lines 4-6 show the instantiation of the different data augmentation components that will be used in the pipeline (a rehydration component to collect additional information from the tweets, a location decoder based on Nominatim, and a sentiment classifier). Lines 9-11 add the components to the demographer object, creating the enrichment pipeline. Finally, line 14 runs the entire pipeline on the data, generating the enriched dataset.

We anyway guarantee the flexibility to allow new components to be implemented. A Component (Listing 2) is a simple abstract class that can be easily inherited and implemented: introducing a custom classification pipeline requires only adding a custom classifier to the pipeline, which inherits this class and implements the methods that handle inputs, outputs and the method to run the inference on the data.

```python
class Component(ABC):

    def __init__(self):
        self.outputs = self.outputs()

    @abc.abstractmethod
    def outputs(self):
        pass

    @abc.abstractmethod
    def inputs(self):
        pass

    @abc.abstractmethod
    def infer(self, *args):
        pass
```

Listing 2: The implementation of the Component abstract class. ‘inputs‘, ‘outputs‘, and ‘infer‘ are all abstract methods that have to be implemented by the inheriting classes.

Inputs and outputs are exploited by Demographer to handle control over the chain of possible components that can be added. A component cannot be added to a pipeline if it requires inputs that are not available in the original data, or that are not generated by previous components. For the sake of providing people with a simple system to extend, the current implementation of Twitter-Demographer represents these variables as lists of strings representing names of columns in data. As a next step, we will improve the current implementation by adopting a pure OOP point of view (i.e., inputs and outputs will turn into interfaces, with configurable parameters).

Listing 3 shows an example of an implemented classifier; this is similar to how we implemented some of our components. However, we report it also to show that this part of the pipeline can be used by interested researchers as an example of code to extend to support custom behaviors in Twitter-Demographer.

Twitter-Demographer saves the intermediate computation steps, right after each component has been executed, to handle down-streaming unexpected errors (e.g., lost internet connection). In those situations, the computation can be restarted from checkpoints.

4 Components

Twitter-Demographer is a container of components and can be extended as they are provided by the community. Some components come with an automatic caching logic, especially when the component relies on external services with a limited request rate (e.g., public API accessed with free accounts with limited requests). For example, the localization component implements a caching mech-

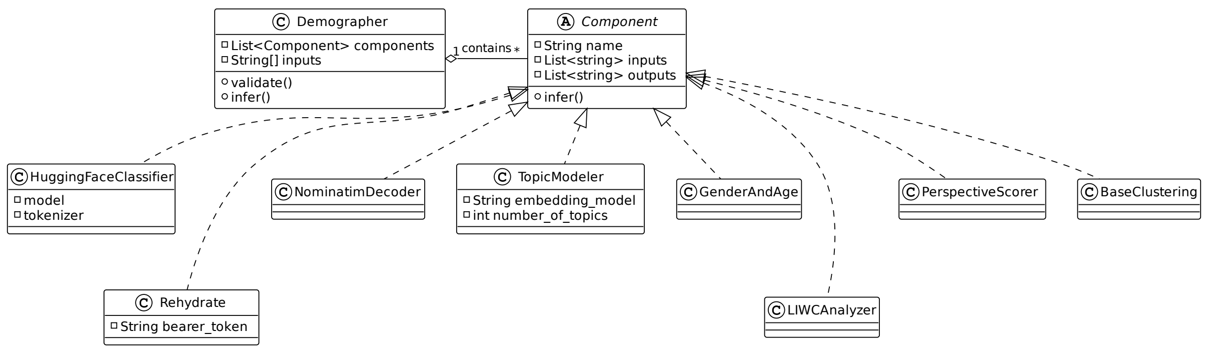


Figure 1: The UML class diagram of the current Twitter-Demographer setup. *Demographer* is the main class that handles the execution of the different *Component*s. *Component* is an abstract class that defines required inputs and produced outputs, as well as an abstract *infer()* method that has to be implemented by its subclasses. Many of the available implementations of the Component class are reported in the UML diagram; the others have been removed to keep the image self-contained.

```
class UserClassifier(Component):

    def __init__(self, model):
        super().__init__()
        self.m = model

    def outputs(self):
        return ["sentiment"]

    def inputs(self):
        return ["text"]

    def infer(self, data):
        return {"sentiment" :
            self.m.predict(data["text"])}
```

Listing 3: A user-defined component for sentiment classification of tweets. Users add their own classifiers to the pipeline by wrapping them inside the Component abstraction.

anism to avoid repeating requests with the same labels, saving requests. The current version of Twitter-Demographer is shipped with the following main components wrapped inside:

Rehydrate Many datasets are released as lists of tweet IDs. This is a requirement that has to be met to release data without violation of Twitter policies. Our Rehydrate Component is based on Twitter API v2. This component allows reconstructing tweets in batches of 100 per request. It automatically waits when Twitter API Rate Limits are reached and starts again when those limits have been reset.[4] This component handles the retrieval of all the information that can be collected on Twitter from the single tweet ID. It requires a Twitter API key.

[4]This is a property we inherit from Tweepy https://www.tweepy.org/

Nomimatim Decoder (Open Street Map) We use Open Street Map as our main source for the localization step, which is the reconciliation of user-written locations to real locations. In our context, we make use of the location present in Twitter profiles. This process is generally less precise than the geolocation given by Twitter, but it also greatly increases the recall as users often fill this field in their profile. This localizer outputs the detected country and city.

Since there is currently no evaluation on the accuracy of this method for Twitter data, we predict locations from the dataset from (Basile et al., 2019) to check how many times the country predicted by the localizer was correct. We asked a single human annotator to annotate 300 samples from this dataset: the prediction is annotated as valid (1) if the country predicted by the model is effectively the one that can be inferred from the user-written location; otherwise, the prediction is annotated as wrong (0) in two occasions: if no country can be inferred from the user location, but the localizer still returns something, or when a country can be inferred but the localizer returns nothing. Table 1 shows examples of user locations with the predicted country and the label assigned by the annotator. The final accuracy of these predictions is 0.85, suggesting that the localizer is reliable enough for this kind of data. We also repeated the experiment by considering city annotations, observing a final accuracy of 0.80.

By default, this component relies on the publicly available Nominatim service,[5] which comes with a rate limit of 1 request per second; however, the

[5]https://nominatim.openstreetmap.org/

Location	Predicted	Score
Florida, USA	United States	1
Regno Unito	United Kingdom	1
120 countries	United Kingdom	0

Table 1: Examples of annotations from the localizer. In the second example, the user location was written in Italian, but the model was nevertheless able to predict the correct country. In the third example, the localizer should not have predicted something, since *120 countries* is not a country. We count this as an error.

component also supports a local installation of the Nominatim service as its backend.[6]

HuggingFace Transformer Classifier HuggingFace transformers (Wolf et al., 2020) is now one of the most used libraries in the NLP field. Thanks to the HuggingFace Hub, models can be deployed online and used by everyone. We provide a general wrapper for the HuggingFace Transformer Classifier. This library is based on the recent advancement in NLP related to the introduction of transformer models.

With this wrapper, any classification pipeline already present on the HuggingFace website can be used to classify the data (e.g., Hate Speech detection, Sentiment Analysis, Emotion Detection, and Topic Classifier). Obviously, users need to ensure they are using the correct model and assess the performance of the original works. While this judgment may require specific background knowledge, in the end it comes down to, for example, finding a model for sentiment analysis from the HuggingFace models' catalog,[7] then checking the original publication to ensure the model is reliable for the specific ongoing task. However, once the model has been found it is only necessary to specify the model name in the pipeline to get the predictions.

Word Counters Similarly to Dehghani et al. (2017) we also integrate the support for Linguistic Inquiry Word Count (LIWC) (Tausczik and Pennebaker, 2010) in our application. LIWC provides meaningful linguistic and psychological categories that can be used to analyze text. LIWC is proprietary and will require interested users to buy the dictionary.

We also integrate Empath (Fast et al., 2016), an open-source tool to analyze text across different lexical categories (similarly to what LIWC does). The author shows that for the same categories, Empath correlates with LIWC.

Toxicity Classifiers Perspective API[8] is currently one of the most reliable classifier for toxicity detection in text. Trained on a proprietary dataset, the results on different dataset show consistent predictive power (with AUC often higher than 0.9). Perspective API offers the annotation for one main label, called textittoxicity that has been used in several other research works (Gehman et al., 2020).

However, the API offers different predictive labels. In our current component, we include the ones suggested by the authors of the APIs, namely *TOXICITY, SEVERE TOXICITY, IDENTITY ATTACK, INSULT, PROFANITY, THREAT*. This API comes with a rate limit of 1 request per second, but it is free; otherwise, users can get an upgraded API key directly from Perspective and use it in the tool.

Gender and Age Predictor Predicting gender and age is very important to understand speaker characteristics better. To this end, Twitter-Demographer also includes a wrapper around the M3 classifier (Wang et al., 2019) that can be used to predict binary gender, age group (i.e., $\geq$40, 30-39, 19-29, $<$18) and identifies if the Twitter account is an organization profile or not.[9] M3 has been shown to be an efficient method to predict demographic variables over Twitter data. This predictor uses information from the profile image, the user name, and the user description to infer the variables.

Figure 2 shows an example of a dataset enriched with sentiment prediction and location.

4.1 Additional Features

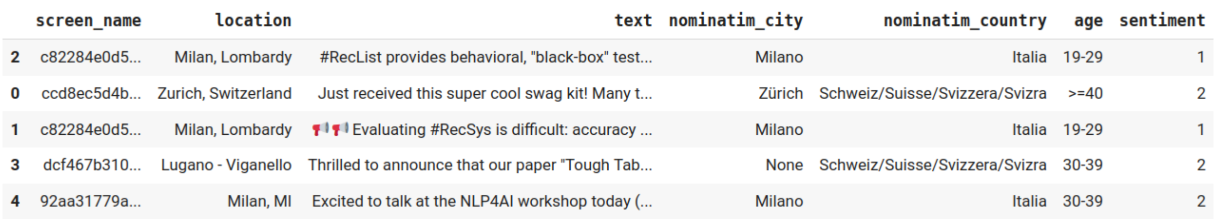
Twitter-Demographer exposes additional and more advanced behaviors through the use of Python decorators. This can be used by more expert users to extend their own pipelines. For example, a common use case is to handle "missing" elements in the pipelines: a geolocalizer cannot be run if the user-written location is not retrieved. This can break the pipeline (i.e., running the Geolocation on `None` generates an error). However, this is often not known at the beginning of the pipeline. This requires writing code to 1) temporarily skip data with missing text, 2) run the classifiers 3) return, to the caller, the entire dataset annotated with the new

[6] A Docker image for deploying Nominatim is available at https://hub.docker.com/r/mediagis/nominatim/.

[7] https://huggingface.co/models

[8] https://www.perspectiveapi.com/

[9] See Section 5 and Section 6 for a discussion of privacy by design and limitations



	screen_name	location	text	nominatim_city	nominatim_country	age	sentiment
2	c82284e0d5...	Milan, Lombardy	#RecList provides behavioral, "black-box" test...	Milano	Italia	19-29	1
0	ccd8ec5d4b...	Zurich, Switzerland	Just received this super cool swag kit! Many t...	Zürich	Schweiz/Suisse/Svizzera/Svizra	>=40	2
1	c82284e0d5...	Milan, Lombardy	📢📢 Evaluating #RecSys is difficult: accuracy ...	Milano	Italia	19-29	1
3	dcf467b310...	Lugano - Viganello	Thrilled to announce that our paper "Tough Tab...	None	Schweiz/Suisse/Svizzera/Svizra	30-39	2
4	92aa31779a...	Milan, MI	Excited to talk at the NLP4AI workshop today (...	Milano	Italia	30-39	2

Figure 2: An example of a dataset enriched with sentiment analysis (2 is positive, 1 is neutral), location, and age of the sender information. The 'location' field, extracted with Twitter APIs, has been disambiguated and split into 'nominatim_city' and 'nominatim_country'. Screen names have been hashed (see Section 5 for a discussion on privacy).

property where possible (to not compromise other steps). Twitter-Demographer exposes a simple decorator that automatically applies this kind of filtering (see Listing 4). The same functionality can be useful for pipelines including sentiment classifiers.

```
@not_null("text")
def infer(self, data):

    [...]
    preds = model.predict(data["text"])

    return {"locations": preds}
```

Listing 4: Extending class methods with decorators to support more complex behaviors. The 'not_null' decorator handles skipping null values in the 'text' column so that the pipeline does not break during the flow.

4.2 Additional Resources

Twitter-Demographer is available as a Python package,[10] released under the research-friendly and open-source MIT license. It is also published on the PyPi repository,[11] and can be installed with the `pip` package manager. Tutorial notebooks are released on the GitHub repository. A two minutes video showcasing Twitter-Demographer usage can be found on YouTube.[12]. There is also a longer version of the video.[13]. The package has 57 GitHub stars and more than 6,000 downloads at the time of submission.

[10]https://github.com/MilaNLProc/twitter-demographer

[11]https://pypi.org/project/twitter-demographer/

[12]https://www.youtube.com/watch?v=NYljrfkLnU8

[13]https://www.youtube.com/watch?v=JGWQZVf2Vdw

5 Privacy by Design

Following the recommendations of the EU's General Data Protection Regulation (GDPR) (European Parliament and Council of European Union, 2016), we implement a variety of measures to ensure pseudo-anonymity by design. Using Twitter-Demographer provides several built-in measures to remove identifying information and protect user privacy: 1) removing identifiers, 2) unidirectional hashing, and 3) aggregate label swapping.

At the end of the reconstruction, we drop most of the personal information that we have reconstructed (e.g., tweet ID, profile URLs, images, and so on). The information is anonymized whenever possible, e.g., screen names are replaced with a globally consistent, unidirectional hash code. In this way, we can retain the user-features mapping within the dataset (enabling further analysis, like aggregations), without allowing people to identify Twitter users (at least not without significant and targeted effort). In addition, we randomly swap the set of labels of a subset of the final data, i.e., some labels attached to one instance are transferred to another instance. This procedure reduces the possibility of finding correlations between individual texts and their labels, which reduces its value for model training. However, we expect this use not to be a user priority. On the other hand, swapping does not affect aggregate statistics and the kind of analysis based on them.

6 Conclusions

We are constantly improving this library to support more use cases and models. For example, we are working on making the geolocation independent of third-party APIs like Nominatim, trying to support the download of the Nominatim index instead to query (thus improving speed and mitigating rate limits). We are introducing multiple methods for topic modeling and additional components for text-clustering (Bianchi et al., 2021; Grootendorst, 2022) and hyperparameter optimization tools to find the optimal values for these. We aim to provide a simple interface to address different user needs. While the tool is momentarily focused on Twitter, most of the components that we have defined have a broader usage (e.g., the localization component).

Ethical Considerations and Limitations

Inferring demographic attributes of users has many advantages for both data analysis and social science research, but it has obvious dual-use potential. I.e., ill-intentioned users could abuse it for their own gains. Users might have chosen not to disclose their information on purpose, so inferring them might go against their wishes. Given the "right" tools, we can also infer protected attributes. Moreover, collecting enough demographic attributes can identify real owners of individual users, or at least reduce the number of potential candidates substantially. The latter raises privacy concerns.

As outlined in Section 5, inferring user attributes carries the risk of privacy violations. We follow the definitions and recommendations of the European Union's General Data Protection Regulation for algorithmic pseudo-anonymity. We implement several measures to break a direct mapping between attributes and identifiable users without reducing the generalizability of aggregate findings on the data. Our measures follow the GDPR definition of a "motivated intruder", i.e., it requires "effort" to undo our privacy protection measures. However, given enough determination and resources, a bad actor might still be able to circumvent or reverse-engineer these measures. This is true independent of Twitter-Demographer, though, as existing tools could be used more easily to achieve those goals. Using Twitter-Demographer provides practitioners with a reasonable way to protect anonymity.

Twitter-Demographer does not come without limitations. Some of these are related to the components' precision; for example, the Nominatim decoder can fail the disambiguation - even if it has been adopted by other researchers and services. Users must be aware of these limitations and check the components' performance.

While Twitter-Demographer makes it easy to define a reproducible pipeline, it cannot prevent the fact that tweets might disappear over time. Thus, running Twitter-Demographer on the same data after some months can generate different results due to missing tweets.

Twitter-Demographer wraps the API from (Wang et al., 2019) for age and gender prediction. However, those predictions come at cost of over-generalizing and stereotyping: age ranges are extremely broad (e.g., the senior population is put in the same group "> 40"), and gender is represented as binary (i.e., male/female). To this end, our intent is not to make normative claims about gender, as this is far from our beliefs.

Twitter-Demographer needs both text and user profile pictures of a tweet to make inferences; for that reason, Twitter-Demographer has to include such information in the dataset during the pipeline execution. While this information is public (e.g., user profile pictures), the final dataset also contains inferred information, which may not be publicly available (e.g., gender or age of the user). We cannot completely prevent misuse of this capability, but we have taken steps to reduce the risk and promote privacy by design substantially.

Not all components in Twitter-Demographer are available for all languages. For example, Empath is only available in English. LIWC is instead available in other languages but requires getting access to different dictionaries. The same goes for the availability of components like classifiers, languages like English have more resources than other low-resource ones.

Eventually, Twitter-Demographer assumes that the users are aware of the limits of the components they are using. The use of HuggingFace models, for example, requires users to check if the models are indeed effective on the data of interest: using a pre-pandemic sentiment classifier on more recent data, might overestimate the number of *positive* tweets due to the presence of the word "positive" in messages regarding COVID positivity.

Acknowledgments

DH is a member of the Bocconi Institute for Data Science and Analytics (BIDSA). This work was partially conducted while FB was a member of BIDSA. This project has partially received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 949944, INTEGRATOR).

References

Alexandra Balahur. 2013. Sentiment analysis in social media texts. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 120–128, Atlanta, Georgia. Association for Computational Linguistics.

Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online. Association for Computational Linguistics.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. 2019. SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in Twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 54–63, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Federico Bianchi, Silvia Terragni, and Dirk Hovy. 2021. Pre-training is a hot topic: Contextualized document embeddings improve topic coherence. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 759–766, Online. Association for Computational Linguistics.

Vincenzo Cutrona, Flavio De Paoli, Aljaž Košmerlj, Nikolay Nikolov, Matteo Palmonari, Fernando Perales, and Dumitru Roman. 2019. Semantically-enabled optimization of digital marketing campaigns. In *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*, volume 11779 of *Lecture Notes in Computer Science*, pages 345–362. Springer.

Morteza Dehghani, Kate M Johnson, Justin Garten, Reihane Boghrati, Joe Hoover, Vijayan Balasubramanian, Anurag Singh, Yuvarani Shankar, Linda Pulickal, Aswin Rajkumar, et al. 2017. Tacit: An open-source text analysis, crawling, and interpretation tool. *Behavior research methods*, 49(2):538–547.

European Parliament and Council of European Union. 2016. Regulation (EU) 2016/679.

Ethan Fast, Binbin Chen, and Michael S Bernstein. 2016. Empath: Understanding topic signals in large-scale text. In *Proceedings of the 2016 CHI conference on human factors in computing systems*, pages 4647–4657.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3356–3369, Online. Association for Computational Linguistics.

Maarten Grootendorst. 2022. Bertopic: Neural topic modeling with a class-based tf-idf procedure.

Dirk Hovy. 2015. Demographic factors improve classification performance. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 752–762, Beijing, China. Association for Computational Linguistics.

Dirk Hovy and Diyi Yang. 2021. The importance of modeling social factors of language: Theory and practice. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 588–602, Online. Association for Computational Linguistics.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A. Smith. 2014. A dependency parser for tweets. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1001–1012, Doha, Qatar. Association for Computational Linguistics.

Yitong Li, Timothy Baldwin, and Trevor Cohn. 2018. Towards robust and privacy-preserving text representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 25–30, Melbourne, Australia. Association for Computational Linguistics.

Veronica Lynn, Youngseo Son, Vivek Kulkarni, Niranjan Balasubramanian, and H. Andrew Schwartz. 2017. Human centered NLP with user-factor adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1146–1155, Copenhagen, Denmark. Association for Computational Linguistics.

J. Paul Morrison. 2010. *Flow-Based Programming, 2nd Edition: A New Approach to Application Development*. CreateSpace, Scotts Valley, CA.

Kevin Munger. 2017. Tweetment effects on the tweeted: Experimentally reducing racist harassment. *Political Behavior*, 39(3):629–649.

Daniel Preoţiuc-Pietro, Vasileios Lampos, and Nikolaos Aletras. 2015. An analysis of the user occupational class through Twitter content. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1754–1764, Beijing, China. Association for Computational Linguistics.

Zachary C Steinert-Threlkeld. 2018. *Twitter as data*. Cambridge University Press.

Yla R Tausczik and James W Pennebaker. 2010. The psychological meaning of words: Liwc and computerized text analysis methods. *Journal of language and social psychology*, 29(1):24–54.

Svitlana Volkova, Theresa Wilson, and David Yarowsky. 2013. Exploring demographic language variations to improve multilingual sentiment analysis in social media. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1815–1827, Seattle, Washington, USA. Association for Computational Linguistics.

Zijian Wang, Scott Hale, David Ifeoluwa Adelani, Przemyslaw Grabowicz, Timo Hartman, Fabian Flöck, and David Jurgens. 2019. Demographic inference and representative population estimates from multilingual social media data. In *The World Wide Web Conference*, pages 2056–2067. ACM.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.