Co-funded by the European Union

# D3.2 Report on the functionality of the DE-BIAS tool

| | |
|---|---|
| **Date of submission** | **3 September 2024** |
| **Author(s)** | **Orfeas Menis and Jason Liartis (both ThinkCode)** with contributions from Vassilis Tzouvaras, Panagiotis Tzortzis and Arne Stabenau (all Datoptron) and Jochen Vermeulen (EF) |
| **Reviewers** | Kerstin Arnold (APEF), Kerstin Herlt (DFF), Kristina Rose (DFF) |
| **Dissemination level** | **Public** |

| HISTORY OF CHANGES | | |
|---|---|---|
| **Version** | **Publication/Submission date** | **Author(s)** |
| 1.0 | | Orfeas Menis (ThinkCode) and Jason Liartis (ThinkCode) |

## TABLE OF CONTENT

# 1. Introduction

The DE-BIAS project is focused on fostering a more inclusive and respectful approach to the description of cultural heritage collections and the representation of the stories and histories of marginalised communities. Over the past 18 months, the project has created an AI-powered tool designed to automatically identify potentially harmful and offensive terms in the descriptive metadata of cultural heritage and provide insights into those terms' problematic origins.

While the tool is available in the form of an API and a stand-alone application, which are both described below (see [5. Interoperability with Europeana and support for custom data imports](#)), the work on the integration with the Europeana Core Service and the MINT platform under T3.4 is ongoing at the time of writing this report. Furthermore, the evaluation activities conducted in WP4 are expected to provide additional insights that will help improve the tool's functionality in the future.

The DE-BIAS tool leverages vocabularies that integrate offensive language with contextual information and offer suggestions for more suitable alternatives. By employing AI solutions like the DE-BIAS tool, Cultural Heritage Institutions (CHIs) can enhance their role as repositories of valuable knowledge, ideas, and artefacts while simultaneously becoming more sustainable and aligned with the evolving expectations of modern audiences regarding heritage experiences.

# 2. Development and deployment of the bias detection

The development and deployment of the bias detection tool have required a comprehensive approach that began with defining user requirements through detailed user stories and epics, outlined in D3.1, User stories & requirements. These user stories capture the needs and expectations of CHIs and aggregators, who are the primary users of the tool. By translating these requirements into technical specifications, the foundation has laid for the creation of an AI-assisted bias detection tool. This tool is designed to automatically detect biassed or offensive language in cultural heritage metadata records and to provide contextualisation for the detected terms based on the vocabularies leveraged by the tool. Thus, the tool establishes a basis for mitigating actions and helps ensure that the language used in these records is inclusive and respectful. The development process has involved leveraging advanced Natural Language Processing (NLP) services, including lemmatization, smart string matching, and Named Entity Recognition (NER) and disambiguation, to accurately identify and address problematic terms within metadata.

Once developed to its full extent, the DE-BIAS tool will be integrated into the Europeana Core Service Platform, specifically aligning with the Metis Suite. This integration will allow the tool to function seamlessly within the existing Europeana infrastructure, enhancing the platform's capabilities in promoting inclusivity across cultural heritage collections.[1]

Additionally, the tool has been designed to function as a stand-alone application, enabling CHIs and aggregators to independently detect and address biases in their metadata. Through a connection with MINT, the tool will also facilitate the ingestion of enriched metadata into Europeana, thereby contributing to the ongoing effort to improve the quality and inclusivity of Europeana's digital content.

This dual functionality ensures that the DE-BIAS tool is both versatile and powerful, capable of serving a wide range of users within the cultural heritage sector.

# 3. Requirements specification for the bias detection tool

The user stories and epics outlined in D3.1 provided insights into the personas and scenarios that have shaped the core functionalities of the DE-BIAS tool. These narratives represented real-life situations, workflows, and aspirations of key stakeholders within the project consortium, with a particular focus on aggregators, who are among the primary target audiences for the tool.

To align the technical components - initially outlined in basic terms in the project's Grant Agreement - with the specific use cases, relevant technical requirements have been extrapolated from these stories. These requirements serve as a crucial link between the ideal user experience and the technical specifications that form the backbone of the tool. They are described in more detail in the following chapters, thereby providing a comprehensive blueprint of the DE-BIAS tool.

[1] In line with this, the project partners co-designed an UI component on the Europeana website under T3.5, which integrates the tool's output with the descriptive information displayed on an item page. The implementation of this UI component will be completed in autumn 2024.

# 4. The bias detection tool

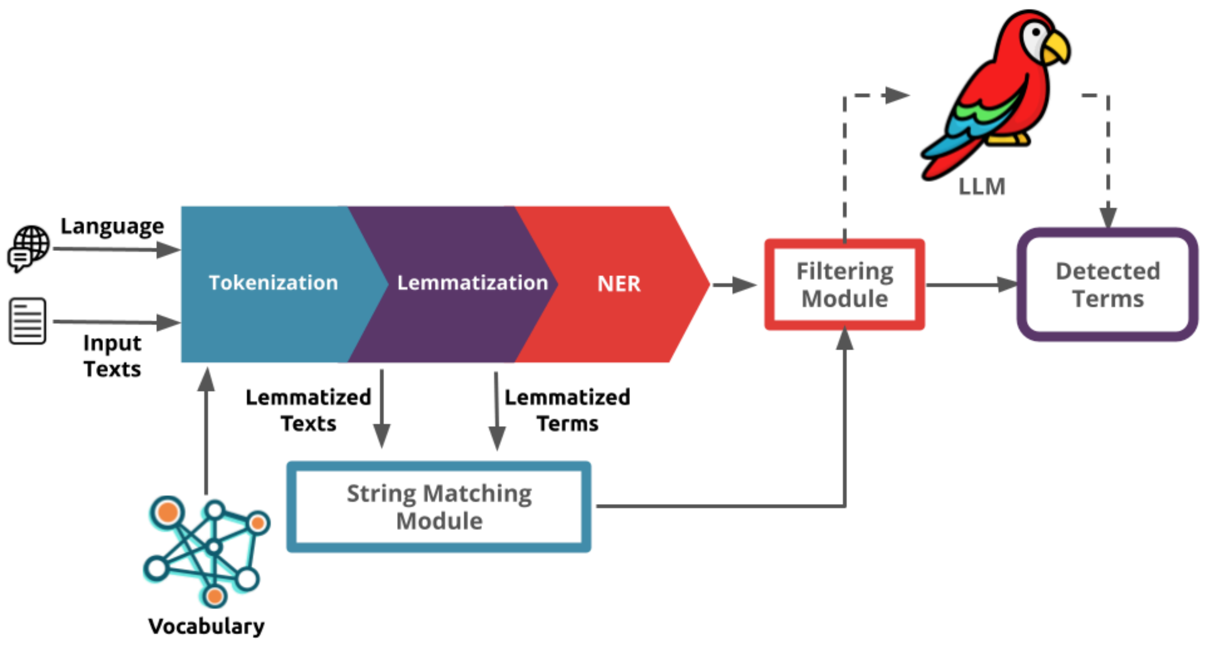


*Figure 1: The workflow diagram of the DE-BIAS detection tool*

The objective of the DE-BIAS tool is to detect contentious terms within the metadata of cultural heritage records, utilising the DE-BIAS vocabulary. Many of these terms are considered derogatory only in certain contexts so the DE-BIAS tool also needs to take the context[2] of each term into account before flagging a specific instance of a term as contentious. The DE-BIAS tool achieves these tasks with a combination of the NLP tools provided by the Stanford Stanza library[3] and the use of a locally deployed Large Language Model (LLM). The main techniques utilised are Lemmatization, NER, and LLM Prompting; the purpose and implementation of each of these techniques are described in detail in subsections 4.1 to 4.3. Additional tools were used for preprocessing the vocabulary and handling compound nouns in German[4]. The main features of the tool are as follows:

---

[2] 'Context' refers specifically to the surrounding textual content where a term is detected, excluding other elements such as digital representations of the described item or additional metadata fields.
[3] Stanza is a Python NLP library which puts several tools, models, and algorithms in a pipeline to conduct text analytics. This includes tokenization, multi-word token (MWT) expansion, lemmatization, part-of-speech (POS) and morphological features tagging, dependency parsing, and NER. See https://stanfordnlp.github.io/stanza/ and Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton and Christopher D. Manning. 2020. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. In Association for Computational Linguistics (ACL) System Demonstrations. 2020. [pdf] for more information.
[4] We are currently investigating the options for implementing a similar functionality for handling compound nouns in Dutch.

- A preprocessing pipeline for external vocabularies stored in a .ttl RDF format. The vocabularies are preprocessed and stored independently of the other components allowing for a plug-and-play approach and the continual development of the vocabularies used with the tool.
- Scripts for modifying and storing the models used by Stanza's NLP pipeline[5], allowing for the addition of rules that handle specific terms to improve the accuracy of the tool.
- An API service that receives requests containing textual records and replies with contentious terms detected in those records specifying their location within the text as described by the DE-BIAS tool API specifications (see chapter 5.1 below).
- Deployment of those modules in the Stanza NLP pipeline that handle the preprocessing of the textual records, including the lemmatization of texts and the recognition of named entities. These modules are deployed on a server with a dedicated GPU[6] accelerator that allows for fast parallel processing of many records. Stanza was chosen due to its selection of state-of-the-art and efficient NLP models in all languages that the DE-BIAS project encompasses and its wide adoption in both research and industry.
- Additional processing of German compound nouns to detect vocabulary terms in their constituent parts.
- Deployment of an independent API service that serves requests to an LLM hosted on a server with a dedicated GPU accelerator. The service is deployed using llama.cpp[7], allowing for easy switching between different models based on performance requirements.
- Development of an LLM filtering component that utilises LLM prompting and the contentious issues described in the DE-BIAS vocabulary to filter detected terms based on the context, in which they appear.
- Archived versions of all the models used are stored on an S3 MinIO object store[8] to ensure consistency and reproducibility.

## 4.1 Lemmatization

A crucial part of our approach is lemmatization, as it allows us to find terms that are present in the text in an inflected form. Lemmatization is the process of determining the lemma, i.e. the canonical or dictionary form of a (set of) word (forms), a more intricate task than simply identifying a word's stem since it is context-sensitive. For example, the word "departed" can

[5] https://stanfordnlp.github.io/stanza/pipeline.html#pipeline
[6] Graphics processing units (GPUs), initially designed for digital image processing and accelerating computer graphics, are increasingly used to perform mathematical calculations - as those present in Artificial Intelligence algorithms at high speed. One area of applying GPUs is the training of neural networks on enormous datasets needed for LLMs.
[7] The llama.cpp project "enable[s] LLM inference with minimal setup and state-of-the-art performance on a wide variety of hardware - locally and in the cloud. See https://github.com/ggerganov/llama.cpp for more information.
[8] MinIO is an object store released under GNU Affero General Public License v3.0 and API compatible with the Amazon S3 cloud storage service. It is used for high performance infrastructures that support machine learning, analytics and application data workloads. See more at https://min.io/.

be a noun with no grammatical inflection, or a verb inflected for tense. It is therefore a challenging task in computational linguistics since it is highly dependent on correctly identifying a word's part-of-speech (POS) within a sentence, which can also be sensitive to the meaning of a sentence and not clearly identifiable by a rigid rule. A relevant example from the DE-BIAS vocabulary is the term “Colored”[9] (debias:t_49_en), as this term is a non-inflected noun but it could be mistaken for the past tense of the verb “to colo(u)r”. Stemming the vocabulary terms and all words present in a document to be processed would result in many false matches since the term “Colored” would not only match with the word “colo(u)red” as in “A colo(u)red photograph” but also with the word “colo(u)ring” as in the sentence “Picasso colo(u)ring his sketches of La Guernica”.

Traditional approaches for lemmatization were dictionary lookup and rule-based systems. Even though these approaches can be very successful for common cases, they can fail for words not present in the dictionary, cases not covered by the rules, or misapplication of the rules due to contextual differences. Machine Learning (ML) has made the adoption of statistical methods possible, which are less rigid than rules and capture patterns that can be extrapolated to different contexts.

For DE-BIAS we deployed Stanza's lemmatizers that employ a combination of rule-based and ML methods to guarantee correct lemmatization for common cases and extrapolation to uncommon ones. Stanza's lemmatizers are also available in many different versions which trade accuracy for processing time. In our tests, the different lemmatizers were processing text at very similar speeds due to the employment of a dedicated GPU (NVIDIA GeForce RTX 4090) as a hardware accelerator for parallel processing, therefore we eventually elected to use the most accurate lemmatizers for DE-BIAS. One important feature of these lemmatizers is that they operate on character level instead of word level. Character-level processing allows the lemmatizer to access the morphology of a word, i.e. analysing its base, potential affixes and suffixes, how the word may behave as POS, how it may be inflected to express number, tense, and aspect. This is especially crucial for processing terms that did not appear in the training data, inflected or not. Since many of the terms present in the DE-BIAS vocabulary are antiquated and do not often appear in more general texts used for training, it is safe to assume that some of them were rare or not present in the data Stanza's lemmatizers were trained on, so character-level processing is a very important feature in our context.

Stanza also allows the addition of custom rules to its lemmatizers, another important tool to handle the terms of the DE-BIAS vocabulary that are rare and Stanza does not handle correctly. We have already identified many such terms, e.g. “chinki” (debias:t_43_en), and have added around 60 custom rules so far, mostly for English, so that the lemmatizer can handle them properly. This is a dynamic process that allows for more rules to be added as the tool is continually tested and improved.

[9] Note that the English part of the DE-BIAS vocabulary currently only uses American English spelling. Depending on the exact difference in spelling, this might have a bigger or a smaller impact on how well the tool detects bias also in British English spellings of the same word.

Stanza reports the following performance levels for its lemmatizers with regard to the languages that concern DE-BIAS[10]:

- Dutch: 95.33%-95.86%
- English: 96.63%-100%
- French: 97.35%-98.50%
- German: 97.23%-98.04%
- Italian: 87.08%-98.35%

The lemmatizers for all five languages used in the DE-BIAS project perform well (close to the maximum 100%), though the detailed performance level will also depend on the test data that these lemmatizers have been trained on compared to the data they are ultimately used on.

## 4.2 Named Entity Recognition

NER is the task of finding spans in a text that refer to named entities such as people, places, organisations etc. Finding named entities in a text is an important step in filtering vocabulary terms detected in the text where these are used in non-contentious ways. For example, the German word “Mohr” is harmful when used to refer to people but it also appears as a last name. NER allows us to detect when “Mohr” appears in text as part of a person’s name and we can filter it from the annotations produced as non-contentious.

It should be noted that the DE-BIAS tool does not cross-reference with any external vocabularies such as Wikidata or similar for NER. Instead, it recognises named entities based on certain patterns and the terms’ context.

We utilise Stanza’s NER models as part of our pipeline and we prefer to use NER to filter annotations before using LLM disambiguation because it is much less computationally demanding and it allows us to reduce the workload of the LLM server.

Stanza reports the following F1-score[11] for its NER models with regard to the languages that concern DE-BIAS. The score varies based on the test data[12]:

- Dutch: 89.20%-94.80%
- English: 88.80%-92.10%

[10] See https://stanfordnlp.github.io/stanza/performance.html (column “Lemmas”) for further details. The figures listed here signify the lowest and the highest performance across all test data within a specific language.
[11] The F1-score is a measure traditionally used in statistical analysis of information retrieval systems. It represents the harmonic mean of the values for precision and recall, i.e. provides a balanced accuracy measure. Alternatively, the Fβ-score provides a measure that allows for weighting and hence emphasising either precision or recall as more important in a specific application context. See https://en.wikipedia.org/wiki/F-score for more.
[12] See https://stanfordnlp.github.io/stanza/ner_models.html for further details. For Dutch, English and German, the figures listed here signify the lowest and the highest F1 score across all test data within each specific language.

- French: 92.90%
- German: 81.90%-85.20%
- Italian: 87.92%

The closer to 100%, the more precise the NER model is (i.e. the fewer false positives it generates) and the higher the recall is (i.e. the fewer false negatives are produced).

## 4.3 LLM disambiguation

As noted by the DE-BIAS vocabulary, some terms can hold a contentious meaning in one context while being neutral outside of that context. One such example is the term “ape”. While it can neutrally describe species of primates, it also has historically been used to compare people of African and Asian descent to these species in order to label them as inferior. The existence of such terms, their inclusion in the DE-BIAS vocabulary, and the diversity of texts that are being processed makes correctly flagging instances of these terms as contentious a much more complicated task. Simply flagging all terms detected by the string matching process, and even filtering them with the use of NER, can potentially flood the annotations produced with false positives due to the contextual nature of many of the terms in the DE-BIAS vocabulary.

For the following explanations, it should be noted that ‘context’ only refers to the textual content directly surrounding the detected term, e.g. the complete title of a cultural heritage object, or its complete description. Digital representations of the described item or other metadata fields, however, are not taken into account when establishing a term’s context, which increases the difficulty of disambiguation when a biassed term is detected in a metadata field that usually contains only little content, such as a subject heading consisting only of one or maybe two words.

Common approaches to the task of disambiguating the meaning of such terms would be converting instances of them to word embeddings from a BERT-like model[13], which could then be used by some sort of classifier that labels them as contentious or non-contentious. The choice of classifier can range from simple ones like a cosine similarity[14] comparison to more complex ones such as a Support Vector Machine[15], but any such algorithm would require tens, if not hundreds, of already annotated examples for each term. Acquiring this volume of high-quality annotated data requires experts and a significant amount of effort, which was not feasible in the restricted timeframe of the DE-BIAS project and the community

[13] Bidirectional Encoder Representations from Transformers (BERT) is a language model that represents texts as a sequence of vectors and has its origins in pre-training contextual representations, including semi-supervised sequence learning, and in generative pre-training. See https://en.wikipedia.org/wiki/BERT_(language_model) for more.
[14] Cosine similarity is a measure of how similar two documents are likely to be, in terms of their subject matter, and independently of the length of the documents. For more details see https://en.wikipedia.org/wiki/Cosine_similarity.
[15] Support vector machines (SVMs) analyse data for classification and for regression analysis, enabling the estimation of relationships between a dependent variable (or the *outcome* / *response* variable) and one or more independent variables (or *predictors* / *features*). For more details see https://en.wikipedia.org/wiki/Support_vector_machine

engagement work leading to the creation of the vocabulary that needed to happen mostly before the work on the tool could start.

A different approach that has gained a lot of traction is the use of Autoregressive (AR) LLMs. AR LLMs are artificial neural networks that are capable of generating human-plausible text. They differ from other LLMs in that their only inherent capability is producing text, possibly as the continuation of an already given piece of text. For example, given the piece of text “Today I went to the...”, a properly trained AR LLM would generate the word “park” or any other word that seems like a plausible continuation of the given text. These models are by default stochastic and do not always generate the same word given a piece of text, so as to have a form of “creativity”, but words that are plausible in the given text have a higher probability to be generated. Even though AR LLMs are intrinsically next-token-predictors, people attribute to them the ability to "understand" language. They have shown incredible knowledge acquisition properties and produce text that respects common facts about the world and contextual meaning. These properties have attracted great attention from the research community and many works have successfully utilised them for tasks other than text generation without having to adapt them. This also made AR LLMs an interesting approach for DE-BIAS where context is essential.

The main method of utilising AR LLMs for different tasks is called “prompting”, or prompt engineering, a technique for steering the output of an AR LLM into a specific direction so that it generates text that is more closely aligned to a given task. A prompt is a piece of text that describes the task that the AR LLM should perform and the process of prompt engineering is discovering which prompt results in the best performance at the given task. In the task of answering multiple choice questions a prompt could be:

> What is the capital of France?
> a) London, b) Paris, or c) Lyon

In the process of prompt engineering, one would study how often the AR LLM generates a valid answer and whether that answer is correct (an invalid answer would be “d”, while an incorrect one would be “a” or “c” as both only represent one part of the question correctly) and then adjust the prompt to achieve better results. AR LLMs, given a well-engineered prompt, have shown excellent performance in many classification tasks, sometimes exceeding that of humans. Many AR LLMs have been released in both a base and an “instruct”, or “chat” version, the latter having been trained in generating valid answers to questions like the above and additionally to other instructive or interactive prompts. The chat versions have generally also been adjusted to better interact with humans, typically by suppressing text that includes hateful, derogatory, or biassed speech, as well as to avoid generating text that can be used in malicious ways, such as instructions on constructing explosives. It has been observed that AR LLMs trained on additional tasks, such as interaction with humans, generally show some performance degradation in other tasks, so even though a chat version of an AR LLM is less likely to generate biassed text, it is not always better at detecting it.

For the purpose of contextual filtering of terms, we deployed an AR LLM locally and developed appropriate prompts for each language. The AR LLM was deployed on a dedicated GPU accelerator (NVIDIA GeForce RTX 4090) using the llama.cpp framework which allows ease of deployment and testing of AR LLMs from many different families of open-source models. The models we tested were mainly from the Mistral AI[16] and Stable LM[17] families because they have been trained on a much more diverse set of multilingual data, compared to others such as Llama.

Some of the models we wanted to test, such as Mixtral-8x7B, did not fit on the memory of our GPU, hence we utilised quantization in order to deploy them. Quantization is the process of converting the parameters of a Deep Neural Network from a large numeric format, commonly float32 and bfloat16, to a smaller numeric format that utilises fewer bits, even as little as one. The process of quantization always incurs some performance degradation, but current research has not yet found decisive arguments to confirm whether quantizing a large model is better than using a small model in its original form. The GPU resource utilisation of various LLMs was tested and compared at various levels of quantization.[18] A total of 100 requests were sent to each AR LLM, with 5 concurrent clients. The requests were evenly distributed between the clients, i.e. 20 requests per client per AR LLM. The resource utilisation was monitored for the duration of the above test and the results for the maximum GPU processing and GPU memory utilisation are detailed in the following table:

| Model | Inference Speed (tokens/s) | GPU Utilisation | GPU Memory Usage |
|---|---|---|---|
| StableLM 2 1.6B Q2_K | 336 | 87% | 7% |
| StableLM 2 1.6B Q4_K_M | 313 | 88% | 8% |
| StableLM 2 1.6B Q5_K_M | 297 | 88% | 9% |
| StableLM 2 12B Chat Q8_0 | 60 | 95% | 53% |
| StableLM 2 12B Chat fp16 | 36 | 97% | 95% |
| Mixtral 7B Instruct | 58 | 98% | 59% |
| Mixtral 8x7B Q2_K | 103 | 88% | 71% |
| Mixtral 8x7B Instruct Q2_K | 104 | 89% | 71% |
| Mixtral 8x7b Instruct Q3_K_L | 88 | 91% | 99% |

[16] https://mistral.ai/technology/#models
[17] https://stability.ai/stable-lm
[18] See https://github.com/ggerganov/llama.cpp/pull/1684#issue-1739619305 for a detailed description of the quantization level for different parameter types.

The model we found performed the best in terms of quality, and within our compute budget, is Mixtral-8x7 Instruct with 3-bit “K_L” type quantization.

To utilise the LLMs for contextual filtering of terms we constructed prompts that include the potentially contentious term we had detected, the contentious nature of the term, as described by the DE-BIAS vocabulary, the text the term was detected in and a description of the task. The English prompt that had universal success in generating valid answers was the following:

> The term "{term}" can be contentious when used in some contexts. Here is an explanation of why "{term}" can be considered contentious: {context}
> Question:
> Is "{term}" used in a contentious manner in the following text, yes or no?
>
> Text:
> {text}
> Answer:

Analogous versions of the prompt are used in the other four languages of the project.

We also suppress any form of stochasticity in the token generation process of the AR LLMs so as to guarantee reproducibility and reliable results.

## 4.4 Vocabulary preprocessing

The DE-BIAS vocabulary is structured as a Knowledge Graph (KG) and externally available in a .ttl RDF format. Instead of accessing the KG directly and completely, the vocabulary is pre-processed and stored in a more easily and efficiently usable format. The parts of the KG that are relevant for the bias detection tool are the term and the contentious issue description available in the .ttl files with the names ‘%Y%m%d-DE-BIAS-Vocabulary-Terms.ttl’ and ‘%Y%m%d-DE-BIAS-Vocabulary-Issues.ttl’. These are ingested using the RDFLib Python library[19] and information regarding each term's URI, literal form, contentious issue description, and whether disambiguation should be applied are extracted. These are then processed into an intermediary Pandas DataFrame format[20]. This process will remain in place even after the vocabulary's publication on VocBench, with regular updates of both files from VocBench directly.

The literal forms of the terms are processed with the same Stanza modules that are applied to the textual records to achieve alignment as to word tokenization, lemmatization and capitalization. To quickly detect terms that consist of multiple tokens, terms are grouped by their first token, which some terms have in common (e.g. term debias:t_101_en “half blood” and term debias:t_102_en “half breed”), and stored in a hash table associating a token with all terms that start with that token and their URIs. This ensures near-constant time

[19] https://rdflib.readthedocs.io/en/stable/
[20] https://pandas.pydata.org/

complexity for detecting terms in texts, regardless of the current or future vocabulary size. In a separate hash table, the term URIs are associated with the contentious issue description of the term and with an indication as to whether the term requires disambiguation. These objects are then saved as serialised Python objects so that they can be loaded without the need for re-processing each time the tool is set up. This also allows the use of any vocabulary that has been converted into the hash map format, or even the intermediary DataFrame format without the need to modify the tool's code.

## 4.5 Tool pipeline

The overall pipeline is as follows.

When starting the tool, the Stanza models are pre-loaded into the GPU memory using their enriched and archived versions for consistency and reproducibility. The processed vocabularies are also pre-loaded in memory. The AR LLM service has been set up independently.

When requests are received, the submitted records are first processed by the Stanza pipeline using tokenization, lemmatization and NER in parallel, as well as compound noun splitting for any records in German, done using a third-party tool that applies an external dictionary and the Aho-Corasick algorithm for exact or approximate multi-pattern string search[21]. Spans of the text are then matched with the vocabulary terms, if they are identical token-by-token. Exact string matching is used instead of subword matching to avoid false positives, such as matching ‘race’ with ‘trace’ or ‘ape’ with ‘aperture’. Matches are produced by first checking for each token of the record if it is the beginning of a vocabulary term using the vocabulary hash table. If a token of the record matches the starting token of a term, the following tokens of the text must match exactly with the rest of the term to produce a match. When a match is produced with a vocabulary term, information is kept regarding the term URI, and the position of the match within the text.

NER and LLM filtering are then applied to the matches, unless specified otherwise by the API request. NER filtering is applied first, and matches are rejected if they are contained within a named entity that is a person, an organisation, a location, a facility, a geopolitical entity, a product, a historical event, a work of art, or the name of a legislative act. LLM filtering is then applied to matches with ambiguous terms of the vocabulary by sending the appropriate prompt to the LLM service, containing the textual record and information about the term as described in section 4.3. On a positive response by the LLM the match is kept as contentious within the context, and on a negative response it is filtered out. Matches that generate an ambiguous or invalid response are also filtered out, but this part of the pipeline has been made easily configurable and can be switched to keeping matches with an ambiguous or invalid response, or can be added to the API specification as configurable for each request.

[21] https://github.com/repodiac/german_compound_splitter/

After the filtered matches have been obtained, annotations are produced. Each annotation is accompanied by a prefix and a suffix that contain the words surrounding the match, keeping as many words as possible without exceeding 50 characters in either direction. The annotations produced contain the term URI, the text span, the prefix, and the suffix. Then, for each item, the annotations are grouped by the term URI, and some annotations are discarded if their total number exceeds the limit set in the API request (see details below) for each combination of item, detected bias term, and metadata field. I.e. if the API request has defined a limit of 1, but a specific bias term is found twice in e.g. the description of an item, only one annotation for this term within the description will be included in the response. The annotations are structured by the API response specification and are returned.

# 5. Interoperability with Europeana and support for custom data imports

An API interface has been designed and developed around the DE-BIAS tool to facilitate the integration of the tool with the Europeana Core Service Platform and with MINT, as well as the use of the tool as a stand-alone service. The stand-alone version allows data providers to apply the DE-BIAS tool on their data independently before publication and the integration with Europeana and MINT allows the direct application of the tool on the data held in these platforms as part of the established data processing workflows.

## 5.1 The DE-BIAS API

The DE-BIAS API provides a layer of communication with the DE-BIAS tool. It needed to meet some non-functional requirements to be usable at a scale, such as:

- Support sending (at least) 100 records at a time
- Support HTTP compression
- Support Keep-Alive header or otherwise support HTTP/2
- Scale-up server instances to increase overall requests per minute

The tool accepts requests following the pattern below:

```
{
 "@context": {
 <CONTEXT_URI>,
 "@base": "http://data.europeana.eu/item/"
 },
 "type": "Request",
 "params": {
 "limitPerPredicate": <LIMIT>,
 "language": <LANGUAGE>,
 "provenance": <PROVENANCE>
 },
 "totalItems": <TOTAL>,
 "items" : [
 {
 "id": <RECORD_ID>,
 "<FIELD_NAME>": [ <FIELD_VALUE>, ... ]
 ...
 },
 ...
 ]
}
```

Where the meaning of the keywords is the following:

- <CONTEXT_URI>: URI of the JSON-LD context, e.g. "https://www.europeana.eu/schemas/context/edm.jsonld"
- <LIMIT>: the number of mentions to be returned per EDM property / field, e.g. the default setting "1"
- <LANGUAGE>: the language of the metadata, using the appropriate language code according to ISO standard 639-1, e.g. "en" for English
- <PROVENANCE>: a boolean value, i.e. "true" or "false", to inform the tool to return or not the provenance information associated with the annotation, meaning "created" and "creator" fields and any other additional provenance information such as confidence levels
- <TOTAL>: optional field to indicate the total number of items that are part of the request, e.g. "10"
- <RECORD_ID>: the local identifier of the record, in relation to the @base, or otherwise, the complete URI of the record within the data.europeana.eu namespace respectively any other namespace if the API is used as a third party, e.g. "18/RML0341041"
- <FIELD_NAME>: the namespace prefixed name of the EDM property/field, e.g. "dc:title"
- <FIELD_VALUE>: the literal value that was associated with the field, e.g. "Roman Central Committee for the collection and dispatch of gifts to soldiers of the 4th. army"; more than one value may be supplied as an array, separated by comma

Then, the API creates responses following the structure below:

```
{
 "@context": [
 "http://www.w3.org/ns/anno.jsonld",
 "https://www.europeana.eu/schemas/context/edm.jsonld",
 "@base": "http://data.europeana.eu/item/",
 },
 "type": "AnnotationPage",
 "partOf": {
 "type": "AnnotationCollection",
 "total": 100,
 "modified": "2024-01-01T12:00:00Z"
 },
 "items" : [
 <ANNOTATION>,
 ...
 ]
}
```

`Where <ANNOTATION>:`

```
{
 "id": "http://example.org/anno29",
 "type": "Annotation",
 "motivation": "highlighting",
 "body": <VOCABULARY_URI>,
 "target": [
 {
 "source": <RECORD_ID>,
 "selector": {
 "type" : "RDFStatementSelector",
 "hasPredicate": "<FIELD_NAME>",
 "refinedBy" : {
 "type" : "TextQuoteSelector",
 "exact": {
 "@value": <BIASED_TERM>,
 "@language": <LANGUAGE>
 },
 "prefix": <PREFIX>
 "Suffix": <SUFFIX>
 }
 }
 },
 ...
 ]
 ...
}
```

The "prefix" and "suffix" are the text segments preceding and succeeding the highlighted value. They should always be stated unless the highlighted value is at the start or at the end of the analysed value in the data field. The maximum character length of both the prefix and the suffix (separately) should be 50 characters. If the 50th character happens to be in the middle of a word, then that word should not be included so as to stay within the 50 characters maximum. The longest possible prefix and suffix that respect these constraints are always returned.

### 5.1.1 The DE-BIAS API in the Europeana Metis Suite

The DE-BIAS API has been integrated into the Metis Suite by way of a client application that can use the API to perform the detection of biassed terms in content (to be) published on Europeana.eu. This integration was leveraged for two purposes:

1. The processing of existing content from the Europeana dataset. Over 7 million items are being processed this way, and the results will be made available on the Europeana.eu website.
2. The option for Metis Sandbox users to activate the detection of biassed terms for data they have processed in the Sandbox. This option will appear once a Sandbox dataset has finished processing. If the user chooses to trigger it, and after the bias detection has finished, a full report will be made available to the user containing all detected occurrences of biassed terms.

### 5.1.2 The DE-BIAS API in MINT

MINT uses the DE-BIAS API to create annotations. On a chosen dataset the user will be offered a selection of XML path elements. MINT extracts all the literal values from those paths and sends them for analysis to the DE-BIAS API. All reported biases are afterwards converted to the MINT annotation format, upon which the annotation view page in MINT can be used to get insights into which and how many biases have been found. Optionally, the annotations can be applied to the dataset to present them as enrichments inside the XML.

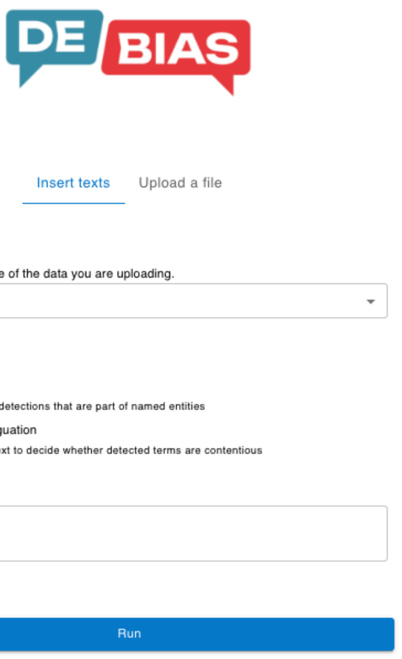
### 5.1.3 Using the DE-BIAS API as a third party

Third parties will be able to use the DE-BIAS API in two main ways. First, the DE-BIAS API will be deployed and made publicly available so that anyone can use it directly. Second, the DE-BIAS tool code, as well as the DE-BIAS API code, will be made publicly available on a GitHub repository under the GNU General Public License[22], so that third parties can deploy the DE-BIAS tool and the DE-BIAS API on their servers. Both the DE-BIAS tool and the DE-BIAS API have been designed so that, at their core, they handle plain text so they can be easily used on various metadata formats with little to no adaptation needed.

## 5.2 The stand-alone version

The stand-alone version is a web application that provides a user interface for interaction with the DE-BIAS tool. It is available here: https://debias-tool.ails.ece.ntua.gr/

This application allows the user to directly include a text via “copy and paste” into the open text field “Enter text” and ask for it to be analysed via the “Run” button.



1 Language *

Select the language of the data you are uploading.

English

2 Operations

NER
Filter out detections that are part of named entities

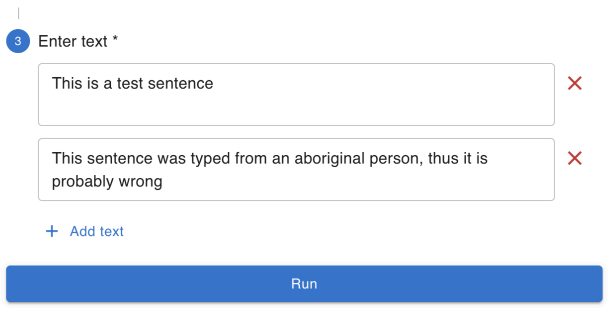
Disambiguation
Use context to decide whether detected terms are contentious

3 Enter text *

*Figure 2: The “Insert texts” tab with default settings (language is English, NER is enabled, disambiguation is disabled)*

[22] See https://www.gnu.org/licenses/gpl-3.0.en.html for details.

It is also possible to add several text segments at once and have them analysed in bulk.

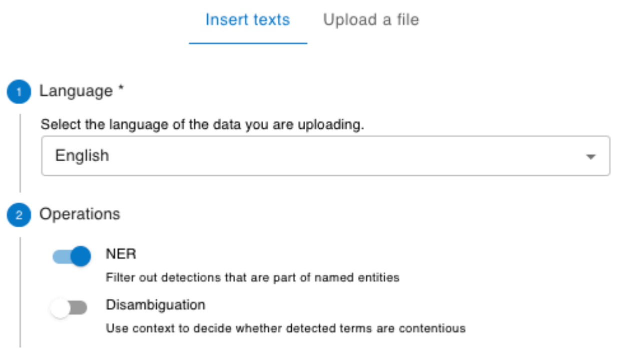

Enter text *

This is a test sentence

This sentence was typed from an aboriginal person, thus it is probably wrong

*Figure 3: Adding several text fragments via “Add text”*

Before running the tool, the user is asked to provide the language of the data to be analysed and can decide whether to enable or disable the NER and disambiguation functionalities of the DE-BIAS tool as described in section 4 via two additional buttons.



Language *

Select the language of the data you are uploading.

English

Operations

NER: on

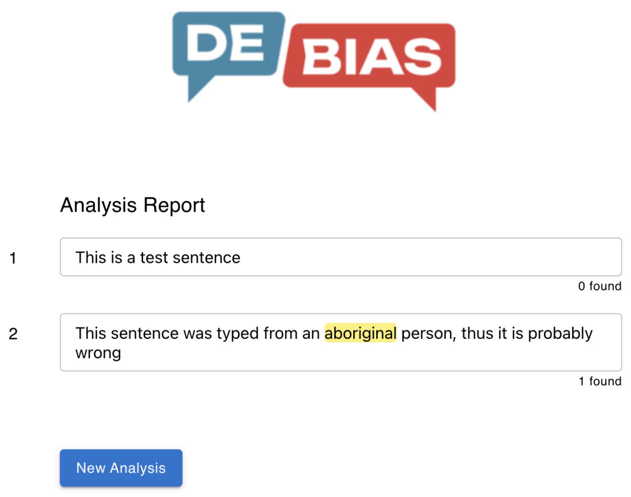
Filter out detections that are part of named entities

Disambiguation: off

Use context to decide whether detected terms are contentious

*Figure 4: Language selection and enabling/disabling of NER and disambiguation*

Once the data input has been analysed, the user is presented with an analysis report highlighting any potentially detected biases.


Analysis Report

1 This is a test sentence

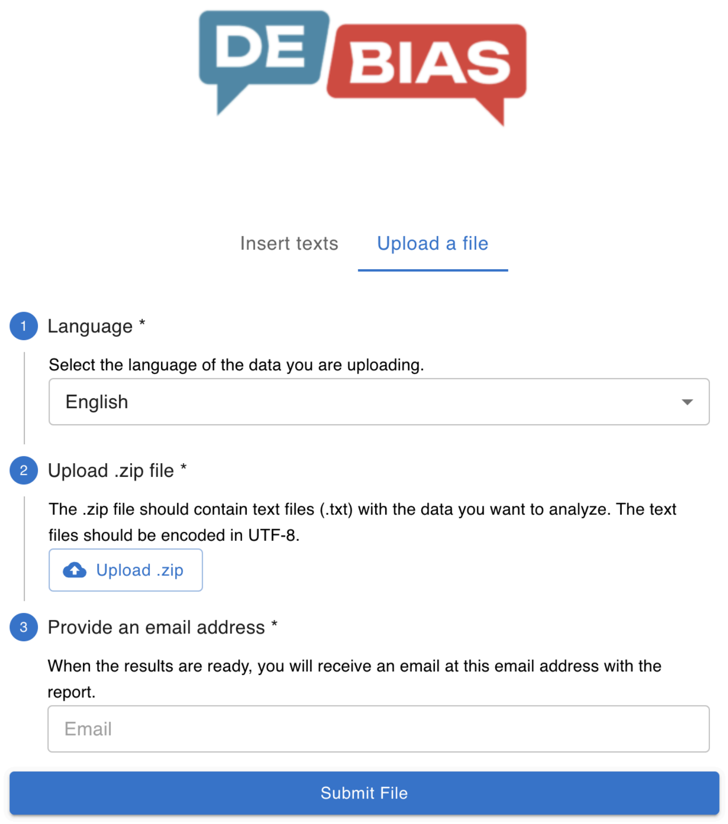
0 found

2 This sentence was typed from an aboriginal person, thus it is probably wrong

1 found

*Figure 5: Analysis report for inserted texts with detected bias highlighted*

Alternatively, the stand-alone tool allows the user to upload a zip file containing text files to submit to the DE-BIAS API and generate some statistics along with the annotations returned from the DE-BIAS tool. Same as when inserting texts directly, the user will have to specify the language of the metadata to be analysed. Furthermore, the user is asked to provide an email address to which the analysis report and the annotations will be sent after processing has finished. The possibility to enable or disable NER and disambiguation when uploading files is not yet available, but will be included in future versions of the tool.



1 Language *

Select the language of the data you are uploading.

English

2 Upload .zip file *

The .zip file should contain text files (.txt) with the data you want to analyze. The text files should be encoded in UTF-8.

3 Provide an email address *

When the results are ready, you will receive an email at this email address with the report.

*Figure 6: The “Upload a file” tab with default settings (language is English)*

In this case, a PDF report is created containing the following information:

General information:

- Number of files (“values”) processed
- Language

Detection statistics:

- Total number of detections/annotations
- The absolute number of files (“values”) processed containing at least 1 detected term
- The average number of detections/annotations per file, and a distribution grouping the analysed files by the number of detections/annotations per file (the x-axis shows

the number of detections per file and the y-axis indicates the number of files with the respective number of detections/annotations)

- The 10 most detected terms or issues of the vocabulary
- The percentage of files with no detection and the percentage of files with at least one detection

The generated PDF report containing all these statistics looks like this:

## Statistical Analysis

After performing debias analysis of the provided data, here are the results:

- Total analyzed values: 7
- Language: en
- Total bias detections: 12
- Total values with at least one detection: 6
- Average number of detections per analyzed value: 1.71

Below you can see the distribution of bias detections on the files:

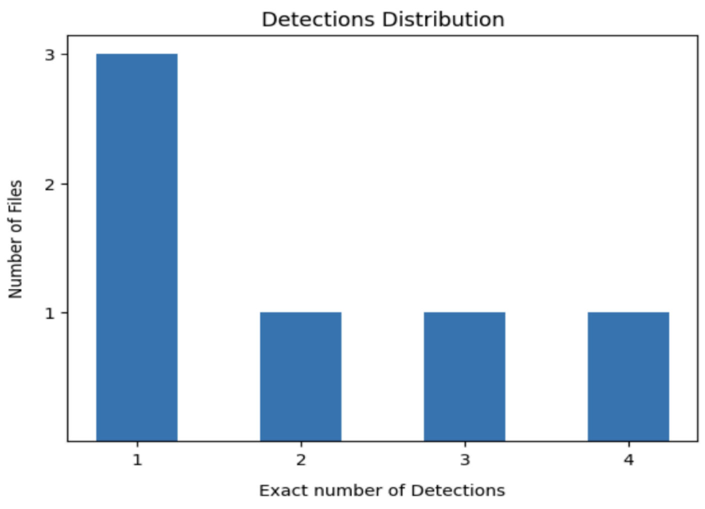


*Figure 7: The distribution of the detections*

Here can see the most frequently detected terms:

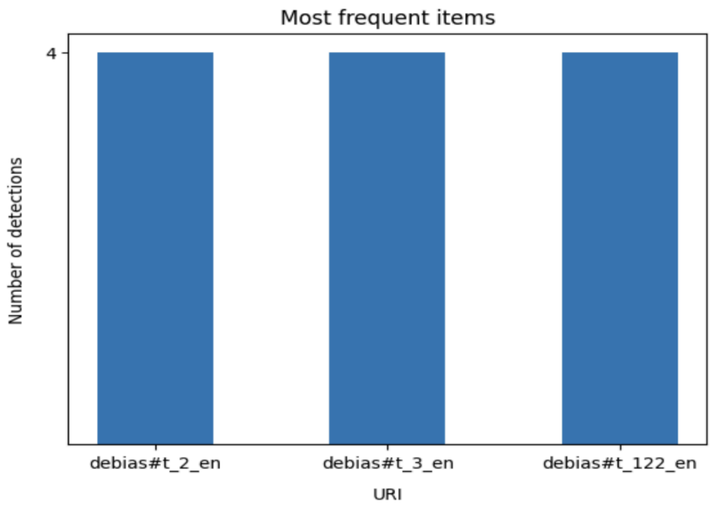


*Figure 8: The most frequent items*

Finally, the percentage of files with at least one detection:

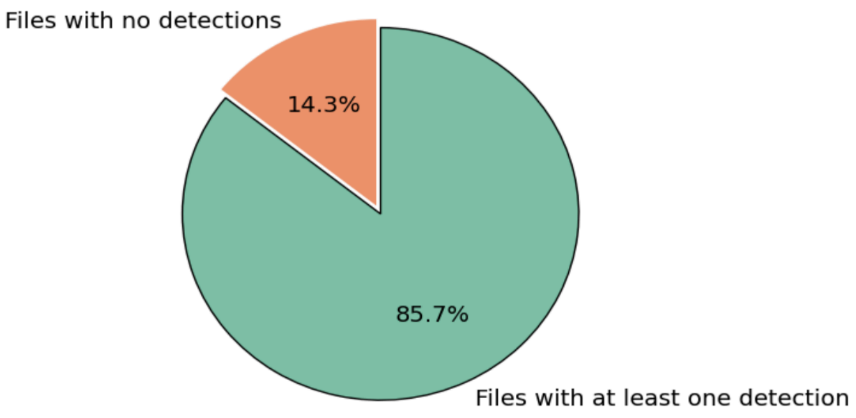


*Figure 9: Percentage of files with and without detection*

As soon as the processing from the tool is finished, the generated report, alongside the annotations produced, is emailed to the email address provided by the user.

The stand-alone version of the tool consists of a frontend and a backend service. The frontend is created using React (https://react.dev/), while the backend is a FastAPI (https://fastapi.tiangolo.com/) Python application using Python 3.10.

## 6. Conclusion

The DE-BIAS tool effectively addresses the challenges of identifying and mitigating bias in cultural heritage metadata. By leveraging advanced NLP techniques, the tool can accurately detect potentially offensive terms and provide contextual insights into their problematic origins. The integration of the tool with Europeana and MINT ensures its accessibility and applicability to a wide range of users within the cultural heritage sector. Furthermore, the tool's ability to function as a stand-alone application empowers data providers to independently assess and address biases in their own collections. Overall, the DE-BIAS tool represents a significant step forward in promoting inclusive and respectful practices in the representation of cultural heritage.

As the DE-BIAS project continues to evolve, the tool will be evaluated and refined. User feedback and insights from evaluation activities will be used to identify areas for improvement and ensure the tool's continued effectiveness in addressing bias in cultural heritage metadata. Additionally, exploring opportunities for collaboration with other relevant initiatives and projects will help to expand the tool's reach and impact within the broader cultural heritage community.

## 7. Acknowledgement

We would like to thank Anna Christaki, Tolis Karageorgopoulos and Krishna Akbar from ThinkCode as well as Kostas Sismanis and Eirini Kladeli from Datoptron for their contributions to the development of the DE-BIAS tool.