Programmable Corpora: Introducing DraCor, an Infrastructure for the Research on European Drama

Frank Fischer[a], Ingo Börner[b], Mathias Göbel[c], Angelika Hechtl[d], Christopher Kittel[e], Carsten Milling[a], Peer Trilcke[f]

a. National Research University Higher School of Economics, Moscow
b. Austrian Centre for Digital Humanities and Cultural Heritage at the Austrian Academy of Sciences, Vienna
c. Göttingen State and University Library
d. Vienna University of Economics and Business
e. University of Graz
f. University of Potsdam

1 Introduction

Although there have been some infrastructural developments of late, the main *modus operandi* in digital literary studies is still to apply a certain research method to an ephemeral corpus. In a best-case scenario, the results are *somehow* reproducible, in a worst-case scenario they are not reproducible at all. At best, there is an openly accessible corpus in a standard format such as TEI, another markup language, or at least TXT. At worst, the corpus is not even accessible, i. e., the research results cannot be questioned.

However, there are signs that this is slowly changing. Some projects provide interfaces that allow for multiple ways of access to corpora. One of these projects is DraCor, an open platform for research on (European) drama, which will be introduced in this paper (accessible at https://dracor.org/ or via its GitHub repositories or its API). DraCor transforms existing text collections into 'Programmable Corpora' – a new term we bring into play with this talk.

2 Building Blocks

2.1 Vanilla Corpora

Similar to the COST Action on European novels (Schöch et al. 2018), the DraCor project seeks to establish a bundle of multilingual drama corpora encoded in basic TEI as basis for digital comparative studies. To date, the platform enables access to a Russian-language (https://dracor.org/rus) and a German-language corpus of plays (https://dracor.org/ger). Similar to Paul Fièvre's collection "Théâtre classique", these corpora are designed as vanilla corpora, which initially contain hardly any special markup beyond the necessary, but are freely available and can therefore be forked, enriched and expanded. To demonstrate that other corpora can be easily linked to the platform, we forked the Shakespeare Folger Corpus and the Swedish Dramawebben corpus and connected it, and all existing extraction and visualisation methods of the platform are readily applicable to the newly added corpora (https://dracor.org/shake and https://dracor.org/swe). Other corpora of dramatic texts are to follow; the only prerequisite is that they are encoded in TEI.

The advantages of a freely available corpus hosted on GitHub are obvious. Not only can the corpus be cloned and loaded directly into an XML database like eXist-db. Using the SVN wrapper from GitHub, the entire corpus can also be downloaded directly, in its current state and without version history if this is not needed:

```
svn export https://github.com/dracor-org/rusdracor/trunk/tei
```

An openly accessible GitHub repository also means that pull requests for error correction are possible and welcome.

2.2 XML Database (eXist-db) and Frontend

DraCor relies on eXist as XML database to process TEI files and to provide functions for researching the corpora. The frontend is built with React (https://reactjs.org/), it is responsive and easily extensible. However, the focus is not on the GUI, but on the API (on the general differences between these two approaches to interfaces cf. Bleier/Klug 2018).

2.3 API

To come close to the ideal and the possibility of applying “all methods to all texts” in a simple manner (Frank/Ivanovic 2018), it takes more than open corpora. The article by Frank/Ivanovic advocates SPARQL endpoints (for which there is also a readily-available app for eXist-db: https://github.com/ljo/exist-sparql).

DraCor offers such endpoint, but also features a rich general API documented and explained via Swagger (https://dracor.org/documentation/api/). In a subarea of corpus philology, the digital scholarly editions, discussions about more proactive use of APIs have already begun (for background information cf. Bleier/Klug 2018), the Folger Digital Texts API may serve as an example (https://www.folgerdigitaltexts.org/api). The advantage of a more modern solution like Swagger is that API queries can be executed live and directly and that the output can be controlled more precisely.

A simple use-case scenario would look like this: using RStudio you can throw a quick glance into a corpus with just a few lines of code, maybe regarding the development of the number of characters in Russian drama between 1740 and 1940, stored in the metadata table (https://dracor.org/api/corpora/rus/metadata). This table, which can be obtained in JSON or CSV format, is read into a Data.Table, whereupon the values of two columns (year of publication and number of speakers) can simply be visualised via ggplot (**Fig. 1**).

This very simple example is able to show the starting point of a decisive structural diversification of the Russian drama landscape. Pushkin’s historical drama “Boris Godunov” (1825),

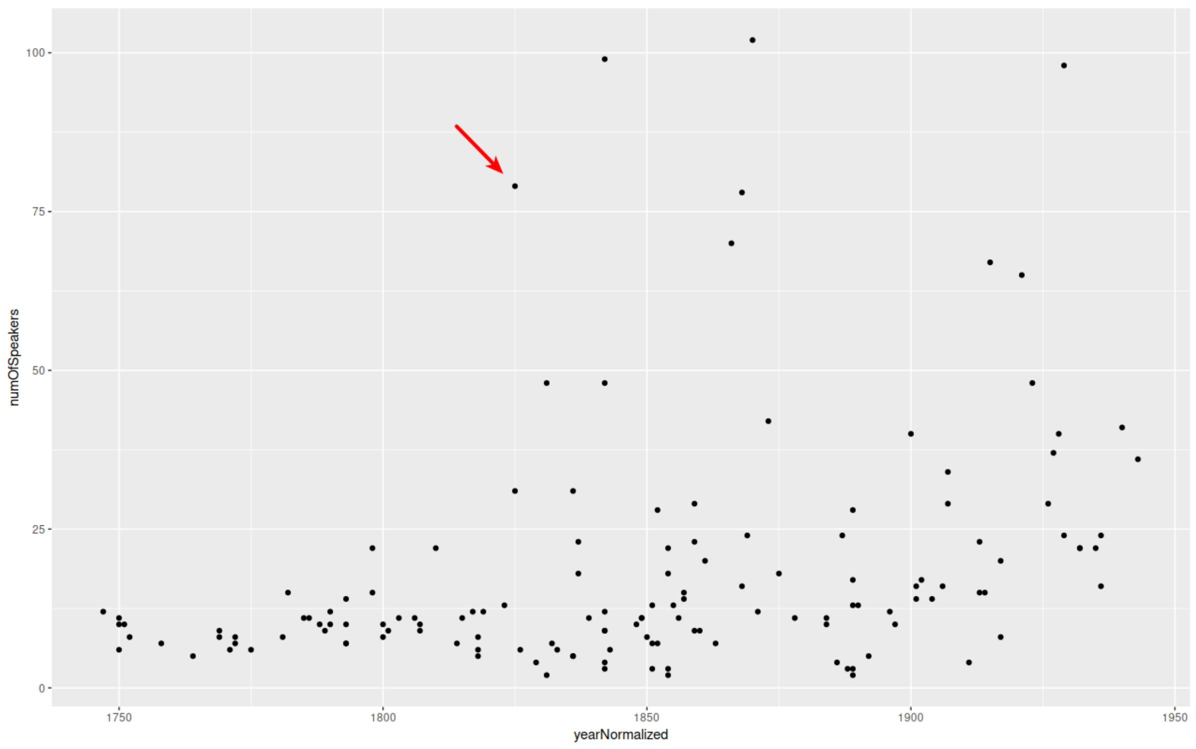

Fig. 1: Number of characters per play in chronological order (source: RusDraCor).

result of his reading of Shakespeare, features speech acts of 79 characters, a number previously unthinkable in Russia drama.

However, the possibilities are not limited to using ready-made API functions. New research ideas always create new needs for easily obtainable and reproducible data and metrics; the API can be extended accordingly, i. e., new research ideas can be implemented centrally in the API layer. This is made even easier by the fact that Apache Ant can be used to rebuild the entire development environment on your own system.

In addition to structural data and metadata, full texts without markup can also be obtained, e. g., if methods such as stylometry or topic modelling are the purpose, i. e., methods that need a “bag of words” and do not require markup.

All in all, the structure and documentation of open APIs makes it much easier to reproduce research results, which up to now has often been a time-consuming (or impossible) process.

2.4 Shiny App

An example of the versatility of the DraCor API is the Shiny App created by Ivan Pozdniakov (https://shiny.dracor.org/). Shiny is a framework based on R, which makes it possible to display interactive visualisations in the browser. The DraCor Shiny App does just that, relying entirely on the DraCor API for data retrieval. Thus, visualisations of the current database can be used for teaching and research purposes, but also for easier data correction.

2.5 Didactics

The formalisation of literary texts, for example via markup, is not self-explanatory. Although the community can rely on some standards, every operationalisation depends on the actual research question. To give an example: if you would like to extract network data based on character interactions in a literary text, you would have dozens of different ways of doing this (e. g., Grayson et al. 2016 test different extraction methods for novels and compare the results). This also applies to plays. In order to sharpen the senses for this in teaching, we developed the tool "ezlinavis" (https://ezlinavis.dracor.org/) and integrated it into the DraCor toolchain. Network data can be extracted from literary texts manually, also to raise the awareness for the contingency of this process, an important preliminary step to the eventual step of operationalisation.

In addition to an approach to the gamification of the process of correcting TEI-encoded corpora (Göbel/Meiners 2016), we also developed a card game for teaching purposes in order to playfully train the understanding of network values (Fischer at al. 2018).

These didactic tools wrapped around the platform are an integral part of the whole project as they are based on the project data and operationalisations. While building the platform, it was important to recognise that data can take several forms and be equally important for research and teaching.

2.6 Linked Open Data

The TEI files contain GND (Integrated Authority File of the German National Library) and Wikidata identifiers for both authors and works. In this way, various data and facts that lie beyond one's own corpus work can be included. Something like an automatically created gallery of authors has a more illustrative character (de la Iglesia/Fischer 2016). But using the same identifiers, we can also determine if a corpus has a regional bias. Via Wikidata, we can easily display the distribution of the authors' places of birth and death on a map (by doing so, we could rule out that our German-language corpus GerDraCor has a regional bias, cf. Göbel/Fischer 2015).

Similarly, the Wikidata ID of the plays can be used to find out where they were first performed (example query: http://tinyurl.com/y9vga68j), i. e., aspects of the performance history can be switched on, even though they are not the focus of the core project and based on data curated elsewhere.

2.7 Infrastructure Instead of Rapid Prototyping

Projects like DraCor seek to provide the digital literary studies with a reliable and extensible infrastructure so that the research community can focus on research questions.

An important conclusion for us was that we would give up the further development of our all-in-one Python script collection "dramavis" (Kittel/Fischer 2014–2018 and Fischer et al. 2017), which we have been developing for four years now. From here on, we would rather devote our time to the API. "Dramavis" followed the idea of rapid prototyping and had to do all by itself, including the preprocessing of data (Trilcke/Fischer 2018), which is not untypical in the

Digital Humanities. The code base has grown quite a bit in the meantime and its maintenance has become difficult and often enough led away from actual research questions.

3 Outlook

In allusion to the project “ProgrammableWeb” – which maintains a database of open APIs and whose slogan is: “APIs, Mashups and the Web as Platform” (accessible at https://www.programmableweb.com/) – we propose the term ‘Programmable Corpora’ for research-oriented corpora providing an API.

Programmable Corpora facilitate the implementation of research questions around corpora. It is to be expected that infrastructural efforts of this kind will pay off for the entire community with effects such as those listed by John Womersley in his presentation at the ICRI2018 conference in Vienna: a) dramatically increase scientific reach; b) address research questions of long duration requiring pooled effort; c) promote collaboration, interdisciplinarity, interaction.

There are numerous ways to connect to Programmable Corpora, no matter if you don’t want to code at all and only need a CSV file for Excel or LibreOffice Calc or a GEXF file for Gephi, if you want to research a corpus via its connections to the Linked Open Data cloud or just want to get specific data for your R or Python script without having to worry about the corpus and its maintenance and reproducibility (all this remains an option, though). Programmable Corpora make it easier to decide on which level of the platform your own research process starts.

References

Bleier, Roman; Klug, Helmut W. (2018). Discussing Interfaces in Digital Scholarly Editing. In: *Digital Scholarly Editions as Interfaces.* BoD, Norderstedt, pp. V–XV. URL: https://kups.ub.uni-koeln.de/9094/

de la Iglesia, Martin; Fischer, Frank (2016). *The Facebook of German Playwrights.* URL: https://dlina.github.io/The-Facebook-of-German-Playwrights/

Fischer, Frank; Dazord, Gilles; Göbel, Mathias; Kittel, Christopher; Trilcke, Peer (2017). Le drame comme réseau de relations. Une application de l’analyse automatisée pour l’histoire littéraire du théâtre. In: *Revue d’historiographie du théâtre.* № 4 (2017). URL: https://hal.archives-ouvertes.fr/hal-01811799

Fischer, Frank; Kittel, Christopher; Milling, Carsten; Schultz, Anika; Trilcke, Peer; Wolf, Jana (2018). Dramenquartett – Eine didaktische Intervention. In: *Conference proceedings of DHd2018.* University of Cologne, pp. 397–398. DOI: https://doi.org/10.6084/m9.figshare.5926363

Göbel, Mathias; Fischer, Frank (2015). *The Birth and Death of German Playwrights.* URL: https://dlina.github.io/The-Birth-and-Death-of-German-Playwrights/

Göbel, Mathias; Meiners, Hanna-Lena (2016). Play(s): Crowdbasierte Anreicherung eines literarischen Volltext-Korpus. In: *Conference proceedings of DHd2016.* University of Leipzig, pp. 140–143. URL: http://www.dhd2016.de/abstracts/vortr%C3%A4ge-007.html

Grayson, Siobhán; Wade, Karen; Meaney, Gerardine; Greene, Derek (2016). The Sense and Sensibility of Different Sliding Windows in Constructing Co-Occurrence Networks from Literature. In: *2nd IFIP International Workshop on Computational History and Data-Driven Humanities.* Trinity College Dublin 2016. PDF: http://derekgreene.com/papers/grayson16chdh.pdf

Kittel; Christopher; Fischer, Frank (2014–2018). *dramavis.* Python script collection. Repository: https://github.com/lehkost/dramavis

Schöch, Christoph et al. (2018). Distant Reading for European Literary History. A COST Action [Poster]. In: *DH2018: Book of Abstracts / Libro de resúmenes.* Mexico: Red de Humanidades Digitales A. C. URL: https://dh2018.adho.org/en/?p=11345

Trilcke, Peer; Fischer, Frank (2018). Literaturwissenschaft als Hackathon. Zur Praxeologie der Digital Literary Studies und ihren epistemischen Dingen. In: Martin Huber and Sybille Krämer (eds.): *Wie Digitalität die Geisteswissenschaften verändert: Neue Forschungsgegenstände und Methoden.* (= Sonderband der Zeitschrift für digitale Geisteswissenschaften, 3). DOI: http://dx.doi.org/10.17175/sb003_003