Editors’ Preface

As the general editors of the DH Benelux Journal, we are proud to present our trilateral Digital Humanities research community with the second volume of this Open Access journal. Like last year, we invited authors of accepted conference abstracts to submit full versions of their papers, which were then subjected to a stringent single-blind peer reviewing process. The resulting volume includes research presented at the sixth instalment of our annual conference, which took place at the University of Liège (Belgium) in September of 2019. The theme of the conference was ’Digital Humanities in Society,’ and as you will see in the papers presented here, an exciting diversity of topics and teams were represented. For an in-depth introduction to the theme and these contributions, be sure to check out the preface by our wonderful guest editors Ingrid Mayeur and Claartje Rasterhoff.

Although we just missed our goal of publishing this volume before the start of this year’s conference, we are glad to still be able to present it to you before the end of the summer. As the ongoing global health crisis increasingly took hold of our personal and professional lives, delays on all fronts of the submission, reviewing, editing, and publication process became inevitable. Despite these difficulties, we are amazed with and thankful for the continued efforts of our authors, reviewers, and co-editors, who made it possible for us to provide you with four strong and substantial contributions.

If the COVID-19 pandemic has taught our academic community anything, it is the extent to which the resilience of our research relies upon the development, sustainability, interoperability, and conscientious criticism of our digital tools, technologies, and methodologies. As our mobility is limited, and the digital research (and other) infrastructures we have developed are subjected to an unsolicited stress test, it becomes abundantly clear just how much we depend on the affordances of digitization – and how our need for reliable and complex digital resources will only continue to grow in the future. This extraordinary year has also reiterated that all of these processes require close collaboration among researchers working in a wide variety of contexts — from traditional faculties to our often under-appreciated colleagues in the GLAM sector who have worked non-stop to provide digital collections and data for our community.

With these new developments in mind, we are especially proud of how eager our community was to quickly change gears once it transpired that this year’s in-person DH Benelux Conference in Leiden would need to be cancelled. In an inspiring feat of flexibility, generosity, hard work, and collaboration, a dedicated group of volunteers put together an incredibly successful three-day virtual event that allowed us to keep the discussion alive, in the form of the high-quality intellectual and social gathering we have come to expect from DH Benelux. We hope that we will be able to draw on this experience in the next few months as we solicit submissions for our next, third edition of the DH Benelux Journal.

Until then: stay safe, take care of yourselves and your loved ones, and enjoy reading the engaging contributions the Editorial Board has put together for you in the present volume!

August 20, 2020
Antwerp and Amsterdam

Wout Dillen
Marijn Koolen
Marieke van Erp

Preface by the Guest Editors

Ingrid Mayeur[1] and Claartje Rasterhoff[1]

[1]University of Liège
[2]University of Amsterdam

The sixth *DH Benelux Conference* was held on 11 — 13 September 2019 at the University of Liège (ULiège), Belgium. The event was organised under the auspices of the CIPL (computer centre of the Faculty of Philosophy and Letters,[1] directed by Dr. Björn-Olav Dozo) and the LASLA (Laboratory of Statistical Analysis of Ancient Languages,[2] directed by Prof. Dr. Dominique Longrée). During those three days, the conference brought together over a hundred participants around the theme "Digital Humanities in Society". Starting in 2014, the annual symposium *DH Benelux* aims to stimulate the collaboration between Digital Humanities researchers in Belgium, The Netherlands, and Luxembourg — although it remains open to everyone, including researchers from outside the Benelux[3]. The conference therefore presents an opportunity for the community of *digital humanists* to meet and exchange around intellectual (or even material) nourishment, by introducing their ongoing projects, discussing their results, and testing their tools. Building on a long tradition of dialogues between the humanities and the computer sciences, the University of Liège proudly hosted the 2019 edition. The two research centers involved in the organisation, the CIPL and the LASLA, have indeed long been concerned with the development of Digital Humanities in Belgium. The first was created in 1983, aiming to promote and to coordinate the use of computer science within the Faculty of Philosophy and Letters. The second was founded earlier, in November 1961, and was the first research centre to have studied the classical languages — Greek and Latin — using automatic information processing technologies. In doing so, the LASLA has collected in computer files numerous ancient Latin works, from Plautus to Ausone, as well as texts from classical Greek literature.

The background of the hosting institutes reflects the way in which, initially, Digital Humanities took off by putting computer technology at the service of research in the humanities —- also described as *Humanities Computing*, and illustrated by, for example, McCarty's eponymous book (McCarty, 2005). Digital technologies have since then constantly evolved, making it possible for humanities scholars to take into account new objects of study, to scale up data collection and to present results in new ways. Moreover, under the influence of web technologies and the networking of texts and data — or, more broadly speaking, of the widespread changeover of our societies to the *digital* — research in the humanities has acquired unprecedented possibilities of

[1] https://www.cipl.uliege.be/cms/c_4535714/fr/cipl
[2] http://web.philo.ulg.ac.be/lasla/. LASLA is part of the research unit (UR) *MOndes Anciens*.
[3] See also the website of the event: http://2019.dhbenelux.org/

dissemination and interaction. Digital Humanities research now goes well beyond the use of computer tools in the humanities, by addressing the various ways in which the humanities are impacted by digitization and datafication, and the role they might be called upon to play in an increasingly digital society.

Although their worth is still too often contested, the humanities are involved in the production of heuristic and critical knowledge that enables actions in the social world as well as the very possibility of a democratic debate (Nussbaum, 2010, Small, 2013). Today, this social world, as well as the man-made artefacts that humanities scholars study, is increasingly digital and datafied (Doueihi, 2008). The changing practices of humanistic research under the impact of digital media, as well as the humanities' ability to question the materiality of the underlying digital infrastructures, challenge us to consider the socio-political make-up of Digital Humanities (Bonde Thylstrup, 2019, Mounier, 2018). The 2019 edition of the *DH Benelux* conference was therefore especially interested in research that addresses Digital Humanities in relation to broader societal transformations: whether these involve new forms of knowledge production and consumption such as citizen science and participatory research methods, or relate to processes of digitisation and datafication in society, including ethical and political issues. In that respect, the symposium aimed to open up the debate on how Digital Humanities should position itself in relation to the various institutional policies that fund or request research that engages with big data, artificial intelligence and data visualizations, and that encourage collaborations with both private and public partners.

Keynotes Lectures

The keynote lectures by Tim Hitchcock (University of Sussex) and Helle Strandgaard Jensen (Aarhus University) addressed the theme of the conference head-on. Both lectures addressed what digital processes in knowledge production and consumption mean for present-day humanities scholarship, and they both confronted us with what it means to be a responsible researcher. Strandgaard Jensen did so by bringing cultural theorist and political activist Stuart Hall (1932-2014) into the conversation, and Tim Hitchcock by invoking Sarah Durrant — a 61 year old widow who, in 1871, was charged with stealing two bank notes. Together, Hitchcock and Strandgaard Jensen revisited the notions of the library and the archive, respectively, employing historical and cultural analyses to create awareness of the political economies and technologies that shape our research on all levels. Their message was clear: when it comes to understanding the knowledge ecosystem in which we work and to which we contribute we need to do more and we need to do better.

Money, Morals and Representation. The day Stuart Hall joined my Archives 101 class

Strandgaard Jensen's lecture on digital archiving literacy reflected her efforts to raise the awareness of historians and other researchers with regard to the way in which digital processes invariably impact their work and their disciplines, and to help archival institutions understand the role they play in this process (Strandgaard Jensen, 2020). Indeed, when collections are being digitized they go through a process of remediation and become part of a new cyberinfrastructure – an infrastructure that too many researchers are still too unfamiliar with. When we (re)use the data from digital archives, she argued, we should do it wisely and knowingly – and that includes understanding the political economy and technical designs of digital archives. As Strandgaard Jensen suggested, Stuart Hall's model can help with this, as it demonstrates how meaning is encoded into the cultural products we consume. As such, we can understand digital archives as digital objects that are encoded by librarians and researchers, funded by stakeholders, made accessible by policy makers, developed by web developers and software engineers, and even the technical capabilities and limitations of the medium they are developed in. But archival institutions also to a large extent conceptualize their archives with a specific user in mind — and in the case of digital collections, those users are often not researchers.

In applying Hall's model to the research of digital archives, Strandgaard Jensen theoretically and empirically researches the archive as a medium that gets remediated when its holdings are digitized. How then has the digital transformation of archival holdings and finding aids affected possibilities for data reuse? How can documentation help researchers avoid data misuse? Based on interviews, analyses of policy papers and the front ends of digital archives, and a multidisciplinary literature review, Strandgaard Jensen encouraged us to improve our collaborative practices between humanities researchers and archival institutions, and to invest in teaching digital (archival) literacy. Her lecture also demonstrated the added value of using the notion of the archive to understand and engage with digital infrastructures, and the need for more empirical research that addresses the construction and knowledge organisation of digital archives, and its impact on methodologies.

Visualising the Infinite Archive

In his lecture, Tim Hitchcock posited that our research methodologies have not kept pace with changing technologies, and that as a result, we now struggle to find trends and meaning in the masses of available data. He argued that there is a fundamental problem with the way in which we represent historical data, or more broadly humanities data, on our screens: with the way in which we search for data, and how we interrogate our search results. Hitchcock argued that the prototypical "lonely search box in the middle of the screen" of most of today's search engines symbolizes a tendency to hide information and strip data of its context — whereas it is exactly this dialogue between data and its sources that is key to effective scholarship. The first step in re-imaging humanities research, Hitchcock proposed, is to go back to the old idea of the library, to rethink our relationship with that "machine for knowing", and to acknowledge the power technologies (both old and new) hold in shaping our research. Here a "macroscope" approach (Börner, 2011), can help us re-imagine search, discovery and research, by providing a new form of "radical contextualisation".

The "macroscope", Hitchcock explained, allows you to see an object at all scales at once — from the most distant to the most granular. It thereby attempts to reconfigure the tools to match humanist methods and, at the same time, to reconfigure our representation of the library as an institution that helps us understand the knowledge systems within which we are working. In his presentation of some of the strategies he developed in collaboration with his colleague Ben Jackson, Hitchcock demonstrated how tools for textual and data analysis can be combined to re-invent a visible and visual context for data. In their demo, they positioned the Old Bailey Online dataset, which encompasses accounts of some 197,745 trials held at the Old Bailey in London between 1674 and 1913 in relation to a set of library and archival catalogues, with the purpose to "allow a new 'open eyed' way of working with data of all sorts — to allow macro-patterns and clusters to be identified; while single words and phrases can be fully contextualised”.[4] The value of this approach, then, lies not only in the possibility to combine close and distant reading, but also in using these technologies to expose the limits of our collections, as well as the structures of authority they reflect — and, by extension, the limits of our knowing.

Journal articles

The four articles selected for this issue are based on papers that were presented during the conference. They are of interest with regard to the conference theme “Digital Humanities in Society”, either by providing through digital methods a better knowledge of the past in order to understand current social/cultural events, or by investigating the digital circulation of research objects specific to the Humanities. For the most part, these papers are the result of a collaborative work. An opportunity to demonstrate once again — if this is still necessary — that the Digital Humanities are a lively field that values the collaborative component of research work.

The contribution that opens this journal issue, “The Datafication of Early Modern Ordinances: Text Recognition, Segmentation, and Categorisation”, directly echoes the issue raised by the keynote speakers of the digital valorisation of heritage texts. C. Annemieke Romein (Ghent University/University Rotterdam/KB National Library of the Netherlands), Sara Veldhoen (KB National Library of the Netherlands) and Michel de Gruijter (KB National Library of the Netherlands) report on the challenges they encountered in the *datafication* of a corpus of early modern printed normative texts (*i.e.* public ordinances or *placards*) under the project *Entangled Histories*. It addresses the need for software-based solutions for recognizing the complex Dutch Gothic print, the segmentation of texts compiled in books of ordinance, the creation of relevant categories of texts, and the automation of categorization. Even if this *datafication* serves to improve knowledge of the rules of Federation-State, such a feedback can be read as a sharing of good practices that could be applied to the treatment of similar collections.

Such *datafication* of old texts helps their automated processing and can result in a reevaluation of previously accepted ideas about these corpora. Theories and findings from other disciplines, then, can help scholars make sense of patterns in larger text corpora, as demonstrated in the contribution by Gianluca Valenti (ULiège): “A Corpus-Based Approach to Michelangelo’s Epistolary Language” . In his essay, Valenti mobilizes quantitative methods such as correspondence analysis and correspondence regression on Michelangelo’s entire epistolary corpus — about 500 handwritten letters. He investigates the traces of a language smoothing over time by using the theoretical frameworks of sociolinguistics and the abundant scientific knowledge of the Florentine dialect. The author shows that, although it is commonly asserted that Michelangelo’s epistolary language would be close to the common contemporary language of 16th-century Florence, his letters display a tension between this language and that of the 14th-century Old Florentine tradition. And that from 1530 onwards — the time when Michelangelo reached the status of a public figure — forms from this Old Florentine language became increasingly prevalent.

The contribution of Chris Tanasescu (UCLouvain, Belgium), Diana Inkpen, Vaibhav Kesarwani and Prasadith Kirinde Gamaarachchige (all three from University of Ottawa) entitled “A-poetic Technology. #GraphPoem and the Social Function of Computational Performance” also exploits the opportunities of computational processing of literary corpora with a focus on its social and technical aspects. The #GraphPoem project

[4] `https://oldbaileyvoices.org/`

relies on the hypothesis of a *performative networked sociality of poetry in digital culture* depending on both humans, poems and machines. The project intends to highlight the way in which poetic texts shape their environment and create the conditions for their reception as they are disseminated within digital media. Starting from such an assumption requires us *to go beyond the poetry*, and to investigate how *medial* and *computational* features actively forge the text as a poem in this digital context. The essay's scientific approach is based on a theoretical framework that integrates both the philosophy of Simondon's technique, and the reappropriation of von Uexkull's *Umwelt* concept by J. A. Schwarz. It leads to an algorithmic treatment of a corpus of digitized poems which intends to uncover the network of relationships in which they are intertwined. It also includes a participatory perspective involving the public in interactive digital performances of computational poetry in order to underline the social dimension of the writing-reading process of poetry through digital spaces.

Using network analysis and data processing tools responsibly means integrating concern for transparency, reliability and reproducibility of research results. The article of Julie M. Birkholz (Ghent University) and Albert Meroño-Peñuela (Vrije Universiteit Amsterdam) addresses this issue through the example of knowledge graphs using the Resource Description Framework language (RDF). These graphs are very popular among digital scholars since RDF provides structured/linked data on cultural objects that are readable by both humans and machines. It thus logically paves the way for network analyses. However, the authors point out the complexities encountered in such an approach — especially the risk of *black boxed tools* — and in making it explicit and reproducible. They therefore introduce a *proof of concept* relying on a concrete tool — a publicly accessible Jupyter Notebook that combines popular libraries in RDF data management and network analysis — the relevance of which they illustrate through two concrete case studies.

Acknowledgements

One cannot end this introduction without mentioning the involvement of the teams that made this event possible. As Program Chairs, Susan Aasman (University of Groningen) and Claartje Rasterhoff (University of Amsterdam) developed the scientific framework of the event, ensuring the programming of the panels and the opening conferences. In Liège, Björn-Olav Dozo (Director of the CIPL) and Dominique Longrée (Director of LASLA) were in charge of the local organization, with the help of Ingrid Mayeur and Benoît Morimont for the logistical support. Financial support was provided by the belgian Fund for Scientific Research (F.R.S.—*FNRS*). Finally, we would like to warmly thank all the presenters at the conference, the reviewers of all the abstracts, the *DH Benelux* steering committee, and the editors of this journal for all their work, without which this issue could not have been published.

We end these lines at a time when preventive measures to contain the spread of covid-19 resulted in the cancellation of the *DH Benelux* 2020 live event in Leiden, that was then promptly replaced by a slimmed down online version of the conference. With a thought for all those who have seen their lives turned upside down by the pandemic in one way or another, we hope that the 2021 edition will help us to renew our tradition of scientific exchange in the Digital Humanities community in the Benelux in a fruitful way — a dialogue that we look forward to continuing in person once more.

References

Bonde Thylstrup, N.
2019. *The Politics of Mass Digitization*. Cambridge, MA: The MIT Press.

Börner, K.
2011. Plug-and-play macroscopes. *Communications of the ACM*, 54(3):60–69.

Doueihi, M.
2008. *La grande conversion numérique. [suivi de] Rêveries d'un promeneur numérique*, Points. Paris: Seuil. CR de Pierre Mounier: http://www.homo-numericus.net/article282.html Date de première édition: 2008.

McCarty, W.
2005. *Humanities Computing*. Basingstoke: Palgrave Macmillan. CR: http://www.digitalhumanities.org/dhq/vol/1/1/000001/000001.html.

Mounier, P.
2018. *Les humanités numériques : Une histoire critique*, Interventions. Paris: Éditions de la Maison des sciences de l'homme.

Nussbaum, M. C.
2010. *Not for Profit: Why Democracy Needs the Humanities*. Princeton: Princeton University Press.

Small, H. H.
2013. *The Value of the Humanities*. Oxford: Oxford University Press. CR: http://www.wsj.com/articles/SB10001424052702303650204579374740405172228 Première édition: 2013.

Strandgaard Jensen, H.
2020. Digital Archival Literacy for (all) Historians. *Media History*, 0(0):1–15. Publisher: Routledge _eprint: https://doi.org/10.1080/13688804.2020.1779047.

A Corpus-Based Approach to Michelangelo’s Epistolary Language

Gianluca Valenti

Université de Liège, Unité de recherches "Transitions"

1 Introduction

1.1 Sociolinguistic Background

Because of documentary and literary reasons, the language of Florence is probably the most studied among the Italian dialects. Its historical development, though, has not been as linear as one might think.[1] Indeed, as is the case for many other dialects, it begins to be widely written, also for literary works, during the Late Middle Ages, but—as is *not* the case for many other dialects—it suddenly becomes extremely popular, thanks to famous poets that used it and spread it all around the Peninsula in the 13th century.[2]

It goes without saying that the 14th century permanently sanctions the supremacy of the Florentine language. The ‘Three Crowns’—Dante, Petrarch, and Boccaccio—ennoble it by writing masterpieces the caliber of the *Comedìa*, the *Rerum Vulgarium Fragmenta*, and the *Decameron*.

Thus in Florence, in the subsequent centuries, the spoken language is constantly evolving over time, as is expected to be. However, people from outside Florence, who increasingly need a common code to communicate, take the language of the 14th century as a reference point. Thus, *14F* acquires a greater value than other Italian dialects both because of the literary importance of the texts of Dante, Petrarch, and Boccaccio, and because—unlike other contemporary dialects—it is a written model, which could be studied and learned.

From the end of the *Quattrocento* and throughout the *Cinquecento*, some humanists begin to recommend taking as a linguistic model the *14F* (often called ‘volgar lingua’), disregarding any further development occurred at the spoken level. The first two grammar books are the *Regole grammaticali della volgar lingua* by Giovan Francesco Fortunio (1516) and the *Prose della volgar lingua* by Pietro Bembo (1525)—the latter being by far the most important and most widespread of its kind.[3]

---

[1] In the whole paper, when discussing the language of Florence, I will use the following abbreviations: *14F* = fourteenth-century (also called ‘golden’) Florentine language; *16F* = sixteenth-century (also called ‘silver’—see Castellani (1970): 17) Florentine language.

[2] Among them, one could mention at least Dante da Maiano, Monte Andrea, Chiaro Davanzati, Bonagiunta Orbicciani and Guittone d’Arezzo.

[3] Obviously, I am simplifying a situation that is much more complex than that. Cf. at least Ghinassi (1961), Quondam (1983), Trovato (1991) and, lastly, Valenti (2018) on the problem of the linguistic norm

Surprisingly enough, in the sixteenth century, people from outside Florence are much more prone to learn *14F* than Florentine people themselves. Indeed, the foreigners study *14F* as a completely new language, without any concern for the fact that it is a 'dead' language, which dates back two hundred years. On the contrary, the Florentine people hardly accept to use a language different from the one they speak and write in daily life. Unfortunately for them, the more time passes, the more *14F* is perceived as the language of high society: people from Florence are increasingly required to use it, too, because this is how they are expected to communicate in cultured and educated milieus.

Indeed, in the whole *Cinquecento*, we notice in Florentine texts a tension between the will to keep the contemporary language (= the *16F*), and the need to use the *14F* to communicate with people from outside Florence. In this context, it is therefore of great interest to analyze the historical evolution of the Florentine language throughout the entire 16*th* century.

After performing a correspondence analysis and a correspondence regression on Michelangelo's entire epistolary corpus (about 500 letters), I verified an evolution over time in his use of the language, and I provided a historical explanation to the outcomes of the statistical tests.

1.2 Michelangelo's Language: An Open Question

In this paper, I focus on Michelangelo's epistolary language. On the one side, I have chosen to analyze only letters (leaving aside the many poems written by the sculptor) because the current linguistic studies increasingly show the need to focus mostly on practical texts instead of literary works.[4] Indeed, practical texts are the best choice for linguistic analyses, because they do not aim to be artistic, and frequently belong to authors without any specific literary education (Serianni (2007): 13).

In recent years, the relevance of private correspondence has been quickly perceived by scholars (cf. e.g., Magro (2014): 106). Letters provide a wealth of precious information for linguists, both because they often carry a date and because their language is frequently close to that of ordinary speech, thus offering access to useful data.[5] As is shown, for example, in Culpeper and Kytö (2010): 17, letters are included in the group of speech-related genres and listed among the speech-like typologies. Thus, despite in epistolary writing the interaction takes place asymmetrically over time, communication is similar to that of the oral speech.[6]

Specifically, this research is targeted towards Michelangelo because he can be considered one of those "intermediate individuals, neither erudite not uneducated people,"[7] who are nowadays drawing the attention of scholars. As is well known, since the appearance of the notion of semi-educated writers (Bruni (1978), Bruni (1984)), scholars increasingly discussed the topic, and today there is a strong tendency to consider the writers' level of education as a continuous, rather than as a set of discrete variables

in Italy in the 16*th* century. On Fortunio, cf. the updated bibliographic references in Moreno and Valenti (2017). On Bembo, cf. at least Patota (2017), together with the critical editions made by Dionisotti (1966), Vela (2001) and Tavosanis (2002).

[4] Cf e.g. Antonelli et al. (2014) for a recent overview of the Italian context.

[5] Recently, D'Achille and Stefinlongo (2016): 249 (translation mine) affirmed that the letters "witness the *langue* throughout the ages."

[6] In the last few years, many scientific essays analyze letters from a linguistic point of view: cf. e.g., Fitzmaurice (2002), Nevalainen and Tanskanen (2007) and Auer et al. (2015).

[7] See Testa (2014): 7 (translation mine). About the semi-educated writers (It. 'semicolti'), see at least D'Achille (1994) and Fresu (2014).

(Fresu (2004), Librandi (2004), Bianconi (2013)).

Finally, it is also of interest to study Michelangelo's language because it fits in a complex and much debated epistemological framework, that of the language of arts and artists. Starting with Folena (1951) and Folena (1957), scientific studies on this topic have multiplied, and have involved prominent scholars, from Barocchi (1984) to Nencioni (1995).[8]

Previous scholars have argued that Michelangelo's epistolary language constitutes a representative example of *16F*, and that it does not make use of most of the features that characterize the language of the Three Crowns.[9] Persuasive as this may seem, I suggest that this assumption can be challenged.[10]

The language of Michelangelo's letters testifies to an interesting tension between the contemporary linguistic usage typical of a sixteenth-century man of Florence, and the Old Florentine literary language prescribed by the grammarians. I present in Section 2.2 the results of an investigation conducted so as to determine the extent to which Michelangelo used *14F* and *16F*. Indeed, from my findings, it would seem that the artist was more aware of the Old Florentine linguistic system than was initially assumed, as it appears that, starting from 1530, he was not loath to borrow from it.

2 Methodological Framework

2.1 Limits and Constraints

Before I go any further, I cannot pass over in silence that the boundaries between *14F* and *16F* are less clear than suggested above. As is well known, some of the so-called fourteenth-century linguistic phenomena had already occurred by the end of the *Duecento* and the beginning of the *Trecento*, but the majority of them only appeared in its second half, and became more stable during the following century.[11] Moreover, it is not even clear when exactly those phenomena started to fade. It is probable that some phenomena were still in use in the first half of the sixteenth century, while others had spontaneously evolved, and others still suddenly found themselves in competition with the fourteenth-century linguistic system, which—at some point—replaced them. Accordingly, the labels *14F* and *16F* do not reflect a clear chronological distinction, and each phenomenon should be discussed and evaluated on a case by case basis.

Another issue is that I focus only on the diachronic variable, while I do not take into account the diaphasic variation. Obviously, for a more comprehensive approach, I should distinguish between letters sent e.g. to subordinates or relatives, and letters sent to the pope or to noblemen. The contents of the message hardly are the same, and the overall tone and style can vary significantly from letter to letter. However, because of the high number of documents taken into account, considering uniquely the diachronic variable can lead to interesting results, too.

Two other limits are somehow inherent to such research. First, I analyze only one writer, while—for a comprehensive study of the variation of the Florentine language in the 16th century—many epistolary corpora, written by different authors, should be compared. And second, the open debate about the possibility of determining (and

[8] It goes beyond the objectives of this paper to summarize all references on this topic. See, lastly, Aresti (2019) for recent bibliographical updates.
[9] See, e.g., Ciulich (1973), Nencioni (1965) and, more recently, D'Onghia (2014), D'Onghia (2015), Felici (2015) and Marazzini (2015).
[10] I already argued against this view in Valenti (2017b).
[11] See Palermo (1992) and Manni (1979), two of the most accurate works on the topic.

to what extent) new information about a spoken language from the analysis of its graphic representation, dates back to the creation of the word *scripta* itself (Remacle (1948)). However, as Arcangeli (2011): 10 (translation mine) notes: "if we are willing to formulate some hypothesis on the state of a language between the Middle Ages and the Renaissance, ... we are necessarily forced to base our conjectures on written texts." Indeed, historical linguistics "is not a second-best solution by inevitable necessity, but just the best solution in those areas of study for which oral records are not available, especially when studying long-term developments of language variation and change" (Hernández-Campoy and Schilling (2012): 64).

Then, strictly speaking, I am analyzing only the *scripta*—not the language—of Michelangelo: but analyzing the *scripta* is the only way to get information about his language.

2.2 Results

To collect the corpus, I copy-pasted the texts from Memofonte (2008) to thirteen .txt files, split into time intervals, from 1495 to 1564.[12] At the end of this first step, each file contained all the letters written by Michelangelo over a range of five years.[13]

Then, I deleted all the special characters ".,;:!?'·" and I split the texts, one word per line. At the end of this step, I put every document (= every time interval) into vectors.[14] Subsequently, based on the current bibliography,[15] I selected the major features that differentiate *14F* and *16F*. For every feature, I identified the golden forms (i.e., *14F* forms) and the silver forms (i.e., *16F* forms). I display here all of them:

- **CruoG** = A tonic Latin ŏ, preceded by consonant plus *r*, becomes *uo* in *14F* (ex. Lat. PRŎBA > *pruova*)
- **CruoS** = A tonic Latin ŏ, preceded by consonant plus *r*, becomes *o* in *16F* (ex. Lat. PRŎBA > *prova*)
- **CrieG** = A tonic Latin Ě, preceded by consonant plus *r*, becomes *ie* in *14F* (ex. Lat. BRĚVIS > *brieve*)
- **CrieS** = A tonic Latin ě, preceded by consonant plus *r*, becomes *e* in *16F* (ex. Lat. BRĚVIS > *breve*)
- **schiVG** = Before vowel, /skj/ remains /skj/ (written <schi>) in *14F* (ex. Lat. SCLAVUS > *schiavo* after a passage CL > *chi* in Medieval Latin)
- **schiVS** = Before vowel, /skj/ becomes /stj/ (written <sti>) in *16F* (ex. Lat. SCLAVUS > *stiavo*)
- **lliG** = A plural noun or adjective, ending in -LLI in Latin, ends in *-lli* in *14F* (ex. Lat. CABALLUS, pl. CABALLI > *cavallo*, pl. *cavalli*)
- **lliS** = A plural noun or adjective, ending in -LLI in Latin, ends in *-gli* in *16F* (ex. Lat. CABALLUS, pl. CABALLI > *cavallo*, pl. *cavagli*)

[12] Last viewed: 01.04.2020. This digital edition accurately reproduces the text of the most important critical edition of Michelangelo's letters (Barocchi and Ristori (1983)).

[13] Notice that we do not possess any letter dated between 1500 and 1505.

[14] For all the steps that follow, I used the software *R* (Team (2020)), and the packages "ca", "mclm", "factoextra" and "FactoMineR."

[15] See Migliorini (1955), Castellani (1970), Castellani (2000), Buck and Pfister (1971), Manni (1979), Palermo (1992), Salani (1992), Renzi and Salvi (2010).

- **artG** = The singular masculine definite article is *il* in *14F*
- **artS** = The singular masculine definite article is *el* in *16F*
- **prIVamoG** = The indicative present IV p. ends in -(*i*)*amo* in *14F* (ex. *noi scriviamo*)
- **prIVamoS** = The indicative present IV p. ends in -(*i*)*ano* or *-emo* in *16F* (ex. *noi scriviano, noi scrivemo*)
- **prVIanoG** = The indicative present VI p. ends in *-ano* in *14F* (ex. *loro parlano*)
- **prVIanoS** = The indicative present VI p. ends in *-ono* or *-eno* in *16F* (ex. *loro parlono, loro parleno*)
- **impfVIvanoG** = The indicative imperfect VI p. ends in *-vano* in *14F* (ex. *loro scrivevano, loro mangiavano*)
- **impfVIvanoS** = The indicative imperfect VI p. ends in *-vono* or *-veno* in *16F* (ex. *loro scriveveno, loro mangiavono*)
- **futrG** = The endings of the indicative future have one intervocalic *r* in some verbs in *14F* (ex. *scriverò, scriverai, scriverà*...)
- **futrS** = The endings of the indicative future have two intervocalic *r* in some verbs in *16F* (ex. *scriverrò, scriverrai, scriverrà*...)
- **congG** = The subjunctive present ends in *-a* (I, II and III p.) and *-ano* (VI p.) in *14F* (ex. *che tu abbia, che lui voglia, che loro debbano*...)
- **congS** = The subjunctive present ends in *-i* (I, II and III p.) and *-ino* (VI p.) in *16F* (ex. *che tu abbi, che lui vogli, che loro debbino*...)
- **condG** = The endings of the conditional present have one intervocalic *r* in some verbs in *14F* (ex. *scriverei, scriveresti, scriverebbe*...)
- **condS** = The endings of the conditional present have two intervocalic *r* in some verbs in *16F* (ex. *scriverrei, scriverresti, scriverrebbe*...)
- **trG** = Lat. DE ĬNTRO, DE RĔTRO become *dentro, dietro* in *14F*
- **trS** = Lat. DE ĬNTRO, DE RĔTRO become *drento, dreto* in *16F*
- **senzaG** = Lat. ABSĔNTIĀ becomes *senza* in *14F*
- **senzaS** = Lat. ABSĔNTIĀ becomes *sanza* in *16F*
- **ultimG** = Lat. ULTĬMUS, -A, -UM becomes *ultim-* in *14F*
- **ultimS** = Lat. ULTĬMUS, -A, -UM becomes *utim-* in *16F*

Clearly, here I am only talking of formal variation, significantly different from conceptual variation. The latter refers to the authors' choice between the use of a word (for instance, 'oak') and, e.g., the use of an hyperonymous ('tree'), while the former only concerns the linguistic variation of the same term—such as, in sixteenth-century Florence, the choice between the forms 'senza' and 'sanza', both meaning 'without'. Within this approach, "the downside is that formal variation is only one aspect of a much broader reality, but it is an aspect we claim is worth isolating" (Speelman et al. (2003): 319).

After listing all those features, I ran the corresponding queries all over the thirteen vectors, so to find the total number of occurrences of each feature for each time interval, and then I manually put the outcomes in a single .csv file. I underline that for running the queries, I had to choose between two options. Sometimes, I could search for the exact match. That was the easiest way. So, for example, to find the occurrences of **senzaG** and **senzaS** I could use those scripts: **senzaG <- '\\bsenza\\b'** and **senzaS <- '\\bsanza\\b'**, and thereafter, I calculated the total number of occurrences with the function 'length'. So, for counting the occurrences of the silver form 'sanza' in the letters written between 1495 and 1499, I used the script: **length(conc_re(senzaS, a, as_text = TRUE)$match)**. After that, I recorded the numerical outcome in a separated file.

Sometimes, however, I could not search for the exact match. In this case, for every occurrence, I had to manually check whether the outcome was correct. I did it with the function 'View', like this: **View(conc_re(CrieG, a, as_text = TRUE))**. Below, I explain with a few examples why, under certain circumstances, it was impossible to search for the exact match in a completely automated way.

It can happen that a similar visual outcome represents different grammatical rules: for instance, a software cannot make the distinction between the form 'scriviano' (that belongs to the **prIVamoS** group) and the form 'pregano' (that belongs to the **prVIanoG** group), because—graphically—the two words have the same ending *-ano* (stressed in the first case, unstressed in the second case). The solution that I found, was to write the same script for the two features—**'(ano\\b)'**—, and then disambiguate them on a case by case basis, depending on the context.

Sometimes—as in the case of the **CrieG**, **CrieS**, **CruoG**, **CruoS** groups—the rule applies only to words derived from Latin short vowels (for instance, from Lat. PRĔCARI we get *priego* in *14F* and *prego* in *16F*). If the Latin word has a long vowel, the outcome was a single vowel (and not a diphthong) in both *14F* and *16F* (for instance, from Lat. CRĒDĔRE we always obtain *credo*). Therefore, when the Latin word has a long vowel, the Florentine word is not included in the **CrieG** group. Of course, there was no way that I could automate a procedure to include a word such as *prego* in the **CrieG** group, while leaving aside a word such as *credo*, because the only difference relies on the Latin etymology. The best I could do was to write two scripts such as: **CrieG <- '(?mix)[bcdfgpt] (rie) [^\\b]'** and **CrieS <- '(?mix)[]bcdfgpt] (re) [^0-9] [^0-9]? [^0-9]? \\b'**, and after, check one by one all the results, discarding the words whose outcome *e* did not derive from Ĕ, Ŏ.

Often, I needed to know also the *meaning*—and not only the *etymology*—of the word that I was analyzing. The structure of the words 'stiavo' and 'stiano', for example, is identical, but the former is part of the **stiVS** group, while the latter is not. Similarly, a word like 'begli' is part of the **lliS** group, while 'degli' is not. The four scripts **schiVG <- '(?mix)(schi) [aeou] [^\\b]'**, **schiVS <- '(?mix)(sti) [aeou] [^\\b]'**, **lliG <- '(?mix) [aeiou] (lli) \\b'** and **lliS <- '(?mix) [aeiou] (gli) \\b'** account for more results, if compared to the correct ones, and again, a manual check was needed.

When I was finally done counting the correct number of occurrences of the forms that I was searching for, I put the numeric outcomes (= the number of occurrences of every feature) in a separated file, just as I did with the **length()** function above.

Next, I did chi-squared test on the outcomes, divided by time intervals. The null hypothesis was that the golden and silver forms were randomly distributed over time.

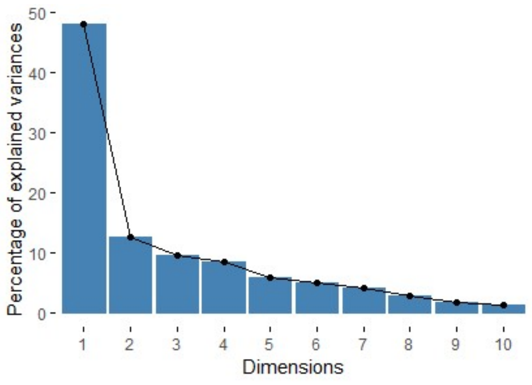


Figure 1: Correspondence analysis scree plot.

I obtained a p-value $<2.2^{-16}$, far below the commonly accepted threshold (0.05).[16]

This means that the outcomes were statistically significant, and consequently, that the use of golden and silver forms varies in a non-random way over the years. But the test does not say in what way non-random choices affected Michelangelo's use of golden and silver forms: that is why I needed to run also the correspondence analysis, "a statistical technique that provides a graphical representation of cross tabulations .... Cross tabulations arise whenever it is possible to place events into two or more different sets of categories."[17]

With the function: **features_ca <- ca(features)** I run the correspondence analysis, and I printed the scree plot (Figure 1).

Given these results, I considered only dimension 1 (time-related), which accounts for 50% of the total.[18] Furthermore, since I was interested in differences among time intervals, I focused on rows (**map="rowprincipal"**, cf. Nenadic and Greenacre (2007)). At that point, I could finally print the plot of the correspondence analysis (Figure 2).

The data and the subsequent plot show a clear cut-off date around 1530: indeed, values consistently diverge before and after this date. On the left side of Figure 2, together with all the time intervals from 1495 to 1530, are grouped most of the silver forms (= *16F*, recognizable by a capital 'S' at the end of their names). On the contrary, on the right side of the plot, together with the time intervals from 1530 to 1564, are grouped most of the golden forms (= *14F*, recognizable by a capital 'G' at the end of the name). Moreover, since the first dimension, corresponding to the horizontal axis, is time-sensitive, I could deduce from the plot a strong separation between silver forms, most of them at the very left side of the plot, and golden forms, most of them—with the only exception of **ultimS**—at the right side.

However, correspondence analysis could be also sensitive to variation different than

[16] The establishment of a threshold has always been a topic of debate; many scholars steadily warned against all possible misinterpretations. In general, if the p-value is much greater or much lower than 0.05, then it is possible to reject (or not) H_0, but if its value is close to 0.05, then all we can do is *not completely reject* H_0. Fisher himself cautioned against a strict exploitation of the threshold as a benchmark for rejecting or accepting H_0: in his words, "if p is between 0.1 and 0.9 there is certainly no reason to suspect the hypothesis tested. If it is below 0.02 it is strongly indicated that the hypothesis fails to account for the whole of the facts. We shall not often be astray if we draw a conventional line at 0.05" (Fisher's quote is taken from Biau et al. (2010): 886b). Furthermore, on "four misconceptions about what a p-value tells us," cf. Nicenboim and Vasishth (2016a) and Nicenboim and Vasishth (2016b).

[17] See Yelland (2010): 1. On correspondence analysis, Greenacre's works are still a great point of departure: see Greenacre (1984), Greenacre (1993) and Greenacre and Blasius (1994).

[18] An extended discussion on the contribution of the axes is in Bendixen (1995).

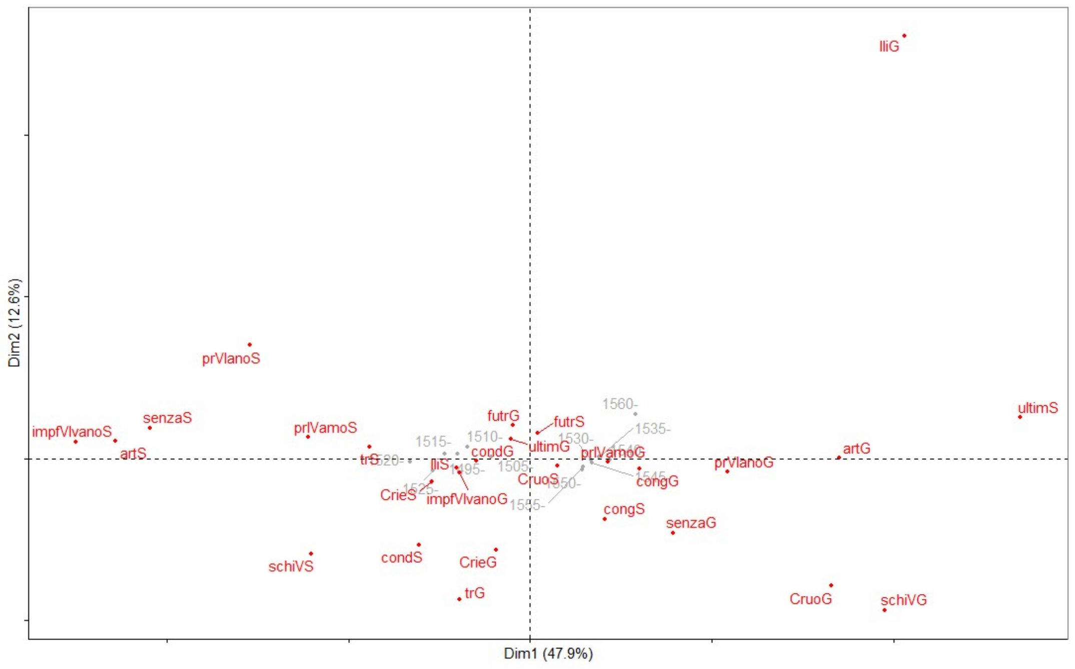


Figure 2: Correspondence analysis.

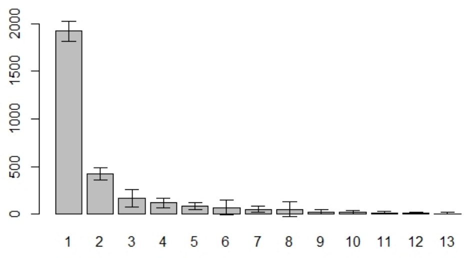


Figure 3: Correspondence regression scree plot.

time, or gold/silver variation, because the features under scrutiny are not completely independent, but come in pairs of alternative variants (*14F* vs *16F* forms). One could argue that in cases like this, the frequencies of the features are not only determined by the writer's preference for the one or the other variant, but also by the overall frequency of the lexical items at hand, as is fully explained in Speelman et al. (2003). To address this issue, I applied correspondence regression, using the *R* package "corregp" (cf. Plevoets (2015)).[19]

I then reshaped the data, so to obtain Table 1. At this point, I performed a correspondence regression of the response variable "feature" in function of the main effect of "time" + the main effect of "measure" + the interaction between "time" and "measure" (cf. Plevoets (2018): 2–3). The plot in Figure 3 shows that dimension 1 (time-related) and dimension 2 (related to the golden and silvery alternation) account for most of the variation (to be precise, 79% of it).

[19] On the other side, Geeraerts et al. (1994), together with Geeraerts et al. (1999), provide a different approach to this issue.

Table 1: Golden and silver forms in Michelangelo's letters

		Time												
Feature	Measure	1495	1505	1510	1515	1520	1525	1530	1535	1540	1545	1550	1555	1560
art	G	1	34	13	11	3	2	31	7	79	143	46	66	38
	S	5	35	38	80	54	36	5	1	5	5	2	12	2
cond	G	3	32	21	35	19	25	16	2	12	37	19	29	12
	S	0	6	0	2	1	3	2	0	0	2	3	1	0
cong	G	0	10	13	12	4	4	7	1	14	35	7	14	4
	S	3	41	14	31	15	13	10	0	27	88	45	23	4
Crie	G	0	3	0	8	3	4	1	0	2	11	6	2	0
	S	3	82	53	82	56	36	8	3	51	82	45	43	6
Cruo	G	0	2	1	1	0	0	1	0	5	14	8	10	0
	S	2	15	9	19	8	7	5	1	17	29	12	9	5
futr	G	12	162	103	143	47	40	30	5	77	196	50	82	49
	S	0	8	9	7	0	0	4	0	2	8	4	6	1
impfVIvano	G	0	5	0	1	1	0	0	0	2	1	0	1	0
	S	0	1	1	3	3	0	0	0	0	0	1	0	0
lli	G	0	0	3	2	0	0	1	0	3	7	3	2	14
	S	0	9	9	11	8	6	1	0	8	17	7	0	2
prIVamo	G	2	0	0	3	1	2	0	0	3	3	4	3	2
	S	1	12	5	8	5	3	0	0	1	6	2	2	1
prVIano	G	0	3	1	3	2	2	4	0	4	11	9	5	4
	S	0	1	2	2	0	0	0	0	0	1	0	0	0
schiV	G	0	0	0	0	0	0	0	0	0	1	0	1	0
	S	1	0	0	0	0	1	0	0	0	1	0	0	0
senza	G	0	2	1	4	7	7	3	1	8	16	8	23	4
	S	0	8	4	2	3	2	0	0	0	0	0	1	0
tr	G	0	1	1	1	3	11	2	0	3	11	2	1	0
	S	0	0	0	2	1	0	0	0	2	0	0	0	0
ultim	G	0	12	18	22	11	3	5	1	3	26	19	13	7
	S	0	0	0	0	0	0	0	0	9	0	0	1	1

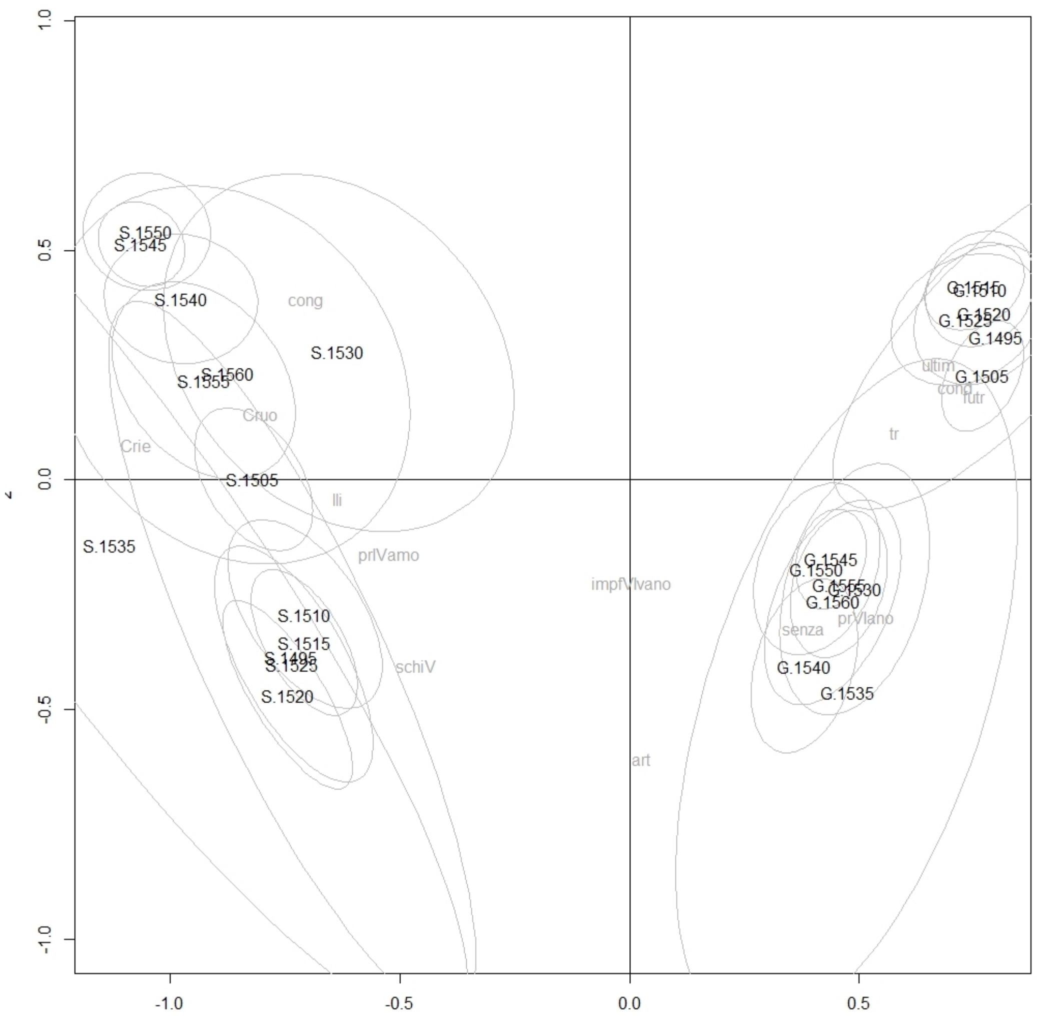


Figure 4: Correspondence regression.

I then plotted the outcomes of the correspondence regression, as in Figure 4. The plot confirms the results previously obtained with the correspondence analysis. In the horizontal axis—that accounts for 65% of the total variation—we notice a strong separation between silver (S.) and golden (G.) forms over time. But also the vertical axis shows a connection between two groups of forms:

1) golden forms, 1495–1530 and silver forms, 1530–1560 (top quadrants);

2) silver forms, 1495–1530 and golden forms, 1530–1560 (bottom quadrants).[20]

The results suggest, one more time, that the key period when Michelangelo started to modify his use of the language is around 1530; moreover, this outcome seems to be independent of the type of lexical items taken into account and their overall frequency.

[20] The only exception is S.1535 in the bottom left quadrant.

3 Historical Interpretation of the Statistical Tests

During his life, Michelangelo repeatedly denied his interest in the contemporary debate on language and grammar, and all previous scholars that studied his texts from a linguistic point of view have insisted that his written style is an excellent example of *16F*. I hypothesize that—despite his repeated claims to be grammatically ignorant—Michelangelo was aware of the existence of manuals prescribing the Old Florentine and may have used some *14F* features more or less deliberately. In this paper, I explored on a quantitative basis Michelangelo's use of the language in his letters.

Among many parameters, I choose to consider *diachrony*. First, I split Michelangelo's letters into documents representing time intervals. Then, I selected the most relevant features that differentiate *14F* from *16F* and I counted their occurrences in the corpus, aiming to see whether there is a difference in his use of the language, and—if so—whether this difference can be related with historical reasons.

The analysis has shown that most of the silver forms are used before 1530, and most of the golden forms are used after 1530, and that their use varies in a non-random way over time. This result is of extreme interest, because the most important Italian grammar book of that time—the *Prose della volgar lingua*, written by the renowned humanist Pietro Bembo—was published in 1525. In that book, Bembo prescribes the use of *14F* (i.e., the golden forms) as a common language for all Italian people.

So far, there was no evidence that Michelangelo ever read the *Prose*. Interestingly enough, my data shows that he consciously started using the *14F* forms a few years after the publication of that book. I argue that Michelangelo was more informed of the contemporary grammatical dispute than we previously thought. It seems reasonable to affirm that in the years following the publication of the *Prose*, Michelangelo read a copy of it, and began modifying his written language, following the *14F* rules prescribed by Bembo. He did so, because in those days he was no more a simple artisan, as he was at the beginning of his life. On the contrary, in 1525–1530 he already became a public figure, and wanted to emancipate from his humble origins: but, to do so, he needed to polish his language from the most marked *16F* phonetic and morphological features, at that time perceived as 'popular'.

The linguistic evidence showing Michelangelo's use of the *14F* forms prescribed by Bembo not only is a reasonable hypothesis, but also perfectly matches with the historical documentation. Indeed, in the 1520s and 1530s, Michelangelo was in Florence, frequenting the *Orti Oricellari* together with his friends Donato Giannotti, Battista della Palla and Antonio Brucioli.[21] They were all devotees to Bembo's ideals, prone to use the Old Florentine language, and they could easily have introduced Michelangelo to that linguistic system. In particular, in those years, Antonio Brucioli was translating Christian texts into vernacular language: he published the New Testament in 1530, the Psalms in 1531 and the Bible in 1532. We also know that in 1529, when Michelangelo was living in Venice, Michelangelo and Brucioli regularly met.[22] Not only then "it is likely that on that occasion, Michelangelo ... has been faced for the first time with the topics of the Protestant Reformation."[23] Moreover, I would like to emphasize that Michelangelo's Venetian stay and the reading of Brucioli's translations may have had some consequences in terms of his linguistic beliefs, too.[24]

[21] On Michelangelo's relationships with other intellectuals, see at least Corsaro (2008).
[22] See Campi (1994): 156.
[23] See Forcellino (2014): 240 (translation mine).
[24] I discussed in further detail the relationship between Brucioli and Michelangelo in Valenti (2017a): 193–195.

Furthermore, in the following years, Michelangelo and Bembo had both stayed at the papal court in Rome. Unfortunately, in the absence of documents witnessing Bembo and Michelangelo friendship, we cannot say much about it, but we are supported by Vasari's words. In the *Life of Michelangelo*, he states:

> The illustrious Cardinal Polo was his close friend, and Michelangelo loved his virtue and goodness. Other friends were Cardinal Farnese and Santa Croce, who later became Pope Marcellus II; Cardinal Ridolfi and Cardinal Maffeo *and Sir Bembo*, Carpi and many other cardinals and bishops and prelates that we do not mention.[25]

Despite there is no evidence of it, the two of them are likely to have discussed grammar and language, and it is possible that Michelangelo showed some kind of interest in the recently printed *Prose della volgar lingua*, the grammar book that was revolutionizing the entire linguistic debate in the whole Peninsula.

Indeed, the preference given by the sculptor in those very years for the use of linguistic features characteristic of the *14F* system may be a reflection of his learned dissertations and his increasing social status, and consequently, might reveal his wish to align his language with the *14F* grammatical rules prescribed by Bembo in 1525. Therefore, the historical documents that witness his frequentations—starting from 1520—with the key players of the sixteenth-century grammatical dispute, confirm and support the results of the correspondence analysis and the correspondence regression.

Likewise, a few years later (1542) Michelangelo asked his friends Donato Giannotti and Luigi del Riccio to amend the language of his poems:

> Sir Luigi, you who have the spirit of poetry, I beg you to shorten and improve one of these madrigals, which at the moment is imperfect, because I must give it to a friend of ours.[26]

As showed in Valenti (2019), he was probably asking for a review of the linguistic features that did not match with Bembo's grammatical rules. This is the last piece of evidence that call into question the old assumption that Michelangelo was unaware of the grammar books prescribing the use of fourteenth-century Florentine language. In fact, his linguistic choices did not always reflect the contemporary use and, sometimes, he was more inclined to employ the archaic forms than we would expect.

Michelangelo Buonarroti once defined himself "grammatically mistaken."[27] Maybe he did not know all the rules of *14F* listed in the *Prose*, as other people of his time did, but this analysis shows that he was far from being completely unaware of them.

Acknowledgements

I am deeply grateful to Pietro Mercuri (Sapienza University, Rome) for his help with the statistical tests. The codes that I used for my analyses are available here: `https://github.com/gianlucavalenti/A-Corpus-Based-Approach-to-Michelangelo-s-Epistolary-Language`.

[25] "Fu suo amicissimo lo illustrissimo cardinale Polo, innamorato Michelagnolo delle virtù e bontà di lui; il cardinale Farnese e Santa Croce, che fu poi papa Marcello; il cardinale Ridolfi e 'l cardinale Maffeo *e monsignor Bembo*, Carpi e molti altri cardinali e vescovi e prelati che non accade nominargli." See Vasari (2011) (last viewed: 01.04.2020. The italic is mine).

[26] "Messer Luigi, voi ch'avete spirito di poesia, vi prego che m'abreviate e rachonciate uno di questi madrigali, quale vi pare il manco tristo, perché l'ò a dare a un nostro amico." See Barocchi and Ristori (1983), vol. IV: 144 (letter n° CMXCVIII).

[27] In his words: "scorrecto in gramatica." Cf. the letter sent to Luigi Del Riccio on February (?), 1544, and preserved in Florence, Archivio Buonarroti, XIII, n. 41 (See Barocchi and Ristori (1983), vol. IV: 177).

References

Antonelli, G., M. Motolese, and L. Tomasin
2014. *Storia dell'italiano scritto. 3 vols. III. Italiano dell'uso.* Carocci.

Arcangeli, M.
2011. L'antico romanesco: gli studi recenti e una vecchia questione. *Bollettino dell'Atlante Lessicale degli antichi volgari italiani*, 4:9–41.

Aresti, A.
2019. *Lingua delle arti e lingua di artisti in Italia fra Medioevo e Rinascimento.* Franco Cesati.

Auer, A., D. Schreier, and R. J. Watts
2015. *Letter Writing and Language Change.* Cambridge University Press.

Barocchi, P.
1984. Storiografia artistica: lessico tecnico e lessico letterario. In *Studi vasariani*, Pp. 134–156. Einaudi.

Barocchi, P. and R. Ristori
1965-1983. *Il carteggio di Michelangelo. Edizione postuma di G. Poggi. 5 vols.* SPES.

Bendixen, M.
1995. Compositional perceptual mapping using chi-squared trees analysis and correspondence analysis. *Journal of Marketing Management*, 11:571–581.

Bianconi, S.
2013. *Italiano lingua popolare. La comunicazione scritta e parlata dei 'senza lettere' nella Svizzera italiana dal Cinquecento al Novecento.* Accademia della Crusca/Edizioni Casagrande.

Biau, D. J. M., B. M. Jolles, and R. Porcher
2010. P value and the theory of hypothesis testing: An explanation for new researchers. *Clinical Orthopaedics and Related Research*, 468:885–892.

Bruni, F.
1978. Traduzione, tradizione e diffusione della cultura: contributo alla lingua dei semicolti. In *Alfabetismo e cultura scritta. Atti del Seminario di Perugia (29-30 marzo 1977)*, Pp. 195–234.

Bruni, F.
1984. *L'italiano. Elementi di storia della lingua e della cultura.* UTET.

Buck, A. and M. Pfister
1971. *Studien zur Prosa des Florentiner Vulgärhumanismus im 15. Jahrhundert.* Fink.

Campi, E.
1994. *Michelangelo e Vittoria Colonna: un dialogo artistico-teologico ispirato da Bernardino Ochino e altri saggi di storia della Riforma.* Claudiana.

Castellani, A.
1967-1970. Italiano e fiorentino argenteo. *Studi linguistici italiani*, 7:3–19.

Castellani, A.
2000. *Grammatica storica della lingua italiana. I. Introduzione*. il Mulino.

Ciulich, L. B.
1973. Costanza ed evoluzione nella grafia di michelangelo. *Studi di grammatica italiana*, 3:5–138.

Corsaro, A.
2008. Michelangelo e i letterati. In *Officine del nuovo. Sodalizi fra letterati, artisti ed editori nella cultura italiana fra Riforma e Controriforma. Atti del Simposio Internazionale (Utrecht, 8-10 novembre 2007)*, Pp. 383–425.

Culpeper, J. and M. Kytö
2010. *Early Modern English Dialogues: Spoken Interaction as Writing*. Cambridge: University Press.

Dionisotti, C.
1966. *Pietro Bembo. Prose della volgar lingua, Asolani, Rime*. UTET.

D'Achille, P.
1994. L'italiano dei semicolti. In *Storia della lingua italiana. Vol. II*, Pp. 41–79. Einaudi.

D'Achille, P. and A. Stefinlongo
2016. Note linguistiche su un corpus di epistolari cinquecenteschi. In *Scrivere lettere nel Cinquecento. Corrispondenze in prosa e in versi*, Pp. 245–262. Edizioni di storia e letteratura.

D'Onghia, L.
2014. Michelangelo in prosa. sulla lingua del carteggio e dei ricordi. *Nuova Rivista di Letteratura Italiana*, 17:89–113.

D'Onghia, L.
2015. Fu vero stile? noterelle su michelangelo epistolografo. *L'Ellisse. Studi storici di letteratura italiana*, 10:135–146.

Felici, A.
2015. *Michelangelo a San Lorenzo (1515–1534). Il linguaggio architettonico del Cinquecento fiorentino*. Olschki.

Fitzmaurice, S.
2002. *The Familiar Letter in Early Modern English: A Pragmatic Approach*. Benjamins.

Folena, G.
1951. Chiaroscuro leonardesco. *Lingua nostra*, 12:57–63.

Folena, G.
1957. Noterelle lessicali albertiane. *Lingua Nostra*, 18:6–10.

Forcellino, A.
2014. *Michelangelo. Una vita inquieta*. Laterza.

Fresu, R.
2004. Alla ricerca delle varietà 'intermedie' della scrittura femminile tra xv e xvi secolo: lettere private di lucrezia borgia e di vannozza cattanei. *Contributi di Filologia dell'Italia mediana*, 18:14–82.

Fresu, R.
2014. Scritture dei semicolti. In *Storia dell'italiano scritto. 3 vols. III. Italiano dell'uso*, Pp. 195–223. Carocci.

Geeraerts, D., S. Grondelaers, and P. Bakema
1994. *The Structure of Lexical Variation. Meaning, Naming and Context*. De Gruyter.

Geeraerts, D., S. Grondelaers, and D. Speelman
1999. *Convergentie en divergentie in de Nederlandse woordenschat. Een onderzoek naar kleding- en voetbaltermen*. Meertensinstituut.

Ghinassi, G.
1961. Correzioni editoriali di un grammatico cinquecentesco. *Studi di filologia italiana*, 19:33–93.

Greenacre, M. J.
1984. *Theory and Applications of Correspondence Analysis*. London Academic Press.

Greenacre, M. J.
1993. *Correspondence Analysis in Practice*. London Academic Press.

Greenacre, M. J. and J. Blasius
1994. *Correspondence Analysis in the Social Sciences: Recent Developments and Applications*. London Academic Press.

Hernández-Campoy, J. M. and N. Schilling
2012. The application of the quantitative paradigm to historical sociolinguistics: Problems with the generalizability principle. In *The Handbook of Historical Sociolinguistics*, Pp. 63–79. Blackwell.

Librandi, R.
2004. Varietà intermedie di italiano in testi preunitari. *Studies in Language*, 8:77–103.

Magro, F.
2014. Lettere familiari. In *Storia dell'italiano scritto*, volume 3, Pp. 101–157. Carocci.

Manni, P.
1979. Ricerche sui tratti fonetici e morfologici del fiorentino quattrocentesco. *Studi di grammatica italiana*, 8:115–171.

Marazzini, C.
2015. La lingua di michelangelo. *Annali aretini*, 23:125–132.

Memofonte
2008. Memofonte. `http://www.memofonte.it/ricerche/michelangelo-buonarroti/`.

Migliorini, B.
1955. Note sulla grafia italiana nel rinascimento. *Studi di filologia italiana*, 13:258–296.

Moreno, P. and G. Valenti
2017. *«Un pelago di scientia con amore». Le Regole di Fortunio a cinquecento anni dalla stampa*. Salerno Editrice.

Nenadic, O. and M. Greenacre
2007. Correspondence analysis in r, with two- and three-dimensional graphics: The ca package. *Journal of Statistical Software*, 20:1–13.

Nencioni, G.
1965. La lingua di michelangelo. In *Michelangelo artista, pensatore, scrittore. Vol. II*, Pp. 569–576. Istituto Geografico De Agostini.

Nencioni, G.
1995. Sulla formazione di un lessico nazionale dell'architettura. *Bollettino d'informazioni del Centro di Ricerche Informatiche per i Beni Culturali*, 5:7–33.

Nevalainen, T. and S.-K. Tanskanen
2007. *Letter Writing*. Benjamins.

Nicenboim, B. and S. Vasishth
2016a. Statistical methods for linguistic research: Foundational ideas – part i. *Language and Linguistics Compass*, 10:349–369.

Nicenboim, B. and S. Vasishth
2016b. Statistical methods for linguistic research: Foundational ideas – part ii. *Language and Linguistics Compass*, 10:591–613.

Palermo, M.
1990-1992. Sull'evoluzione del fiorentino nel tre-quattrocento. *Nuovi Annali della Facoltà di Magistero dell'Università di Messina*, 8-10:131–156.

Patota, G.
2017. *La Quarta Corona. Pietro Bembo e la codificazione dell'italiano scritto*. il Mulino.

Plevoets, K.
2015. *corregp: Functions and Methods for Correspondence Regression*. Ghent University.

Plevoets, K.
2018. Correspondence regression: A tutorial. `https://cran.r-project.org/web/packages/corregp/vignettes/corregp.pdf`.

Quondam, A.
1983. La letteratura in tipografia. In *Letteratura italiana. Vol II: Produzione e consumo*, Pp. 555–686. Einaudi.

Remacle, L.
1948. *Le problème de l'ancien wallon*. Bibliothèque de la Faculté de Philosophie et Lettres de l'Université de Liège.

Renzi, L. and G. Salvi
2010. *Grammatica dell'italiano antico. 2 vols*. il Mulino.

Salani, T. P.
1992. La toscana. In *L'italiano nelle regioni. Lingua nazionale e identità regionali*, Pp. 402–461. UTET.

Serianni, L.
2007. La storia della lingua italiana, oggi. *Bollettino di italianistica n.s.*, 4:5–19.

Speelman, D., S. Grondelaers, and D. Geeraerts
2003. Profile-based linguistic uniformity as a generic method for comparing language. *Varieties Computers and the Humanities*, 37:317–337.

Tavosanis, M.
2002. *La prima stesura delle «Prose della volgar lingua»: fonti e correzioni*. ETS.

Team, R. C.
2020. R: A language and environment for statistical computing. r foundation for statistical computing. https://www.R-project.org/.

Testa, E.
2014. *L'italiano nascosto. Una storia linguistica e culturale*. Einaudi.

Trovato, P.
1991. *Con ogni diligenza corretto. La stampa e le revisioni editoriali dei testi letterari italiani (1470-1570)*. il Mulino.

Valenti, G.
2017a. Le lettere di michelangelo. auto-promozione e auto-percezione nel contesto del dibattito linguistico contemporaneo. *Studi di Memofonte*, 18:182–210.

Valenti, G.
2017b. Towards an analysis of michelangelo's epistolary language: Some remarks. *Italica*, 94:685–708.

Valenti, G.
2018. «de l'uso frequentato si fan norme»: L'italie au xvie siècle, entre normalité et normativité. In *«Modello, regola, ordine». Parcours normatifs dans l'Italie du Cinquecento*, Pp. 323–336. Presses Universitaires de Rennes.

Valenti, G.
2019. Le revisioni linguistiche del riccio e del giannotti alle lettere di michelangelo. In *Lingua delle arti e lingua di artisti in Italia fra Medioevo e Rinascimento*, Pp. 263–290. Franco Cesati.

Vasari, G.
2011. Vita di michelangelo (1568). http://vasariscrittore.memofonte.it/home.

Vela, C.
2001. *Pietro Bembo. Prose della volgar lingua. L'«editio princeps» del 1525 riscontrata con l'autografo Vaticano latino 3210*. CLUEB.

Yelland, P. M.
2010. An introduction to correspondence analysis. *The Mathematica Journal*, 12:1–23.

The Datafication of Early Modern Ordinances

C. Annemieke Romein[1,2,3], Sara Veldhoen[1], and Michel de Gruijter[1]

[1]KB, National Library of the Netherlands
[2]Ghent University
[3]Erasmus University Rotterdam

Keywords: Early Modern Printed Ordinances; Text recognition; Text segmentation; Categorisation; Machine Learning; Annif; Transkribus; Dutch Gothic Print.

The project *Entangled Histories* used early modern printed normative texts. The computer used to have significant problems being able to read Dutch Gothic print, which is used in the vast majority of the sources. Using the Handwritten Text Recognition suite Transkribus (v.1.07-v.1.10), we reprocessed the original scans that had poor quality OCR, obtaining a Character Error Rate (CER) much lower than our initial expectations of <5% CER. This result is a significant improvement that enables the searching through 75,000 pages of printed normative texts from the seventeen provinces, also known as the Low Countries.

The books of ordinances are compilations; thus, segmentation is essential to retrace the individual norms. We have applied – and compared – four different methods: ABBYY, P2PaLA, NLE Document Recognition and a custom rule-based tool that combines lexical features with font recognition.

Each text (norm) in the books concerns one or more topics or *categories*. A selection of normative texts was manually labelled with internationally used (hierarchical) categories. Using Annif, a tool for automatic subject indexing, the computer was trained to apply the categories by itself. Automatic metadata makes it easier to search relevant texts and allows further analysis.

Text recognition, segmentation and categorisation of norms together constitute the datafication of the Early Modern Ordinances. Our experiments for automating these steps have resulted in a provisional process for datafication of this and similar collections.

1 Introduction

Normative rules - from any era - provide very versatile insights into society, as there are many (hidden) layers within such texts. They can, for instance, shed light on ideas on and interpretations of the organisation of society; indicate what troubles society faced; provide insights into communication patterns. In the early modern period (±1500 - 1800) rules were announced by a city crier. He walked through the city or rode a horse to rural villages in order to visit contractually-indicated locations to proclaim new rules and repeat older ones to the local residents. After reading them aloud, the printed texts were fixed to 'well-known places' (e.g. at the church door, trading places (see Figure 1), or at the market square) for people to be able to reread them. Sometimes there was merely a duty to affix the rules, an obligation that was carried out by the city's affixer (Dut. *stadsaanplakker*) (Der Weduwen, 2018). In the mid-seventeenth century about 50% of all urban residents could read in the Republic, so for the remainder of the residents having the new rules read aloud was still very important (Hoftijzer, 2015). The rules had to make sense, so people could remember them by heart. Hence, there is a repetitiveness in the texts – which makes sense given that the 16th and 17th century had an important oral tradition.

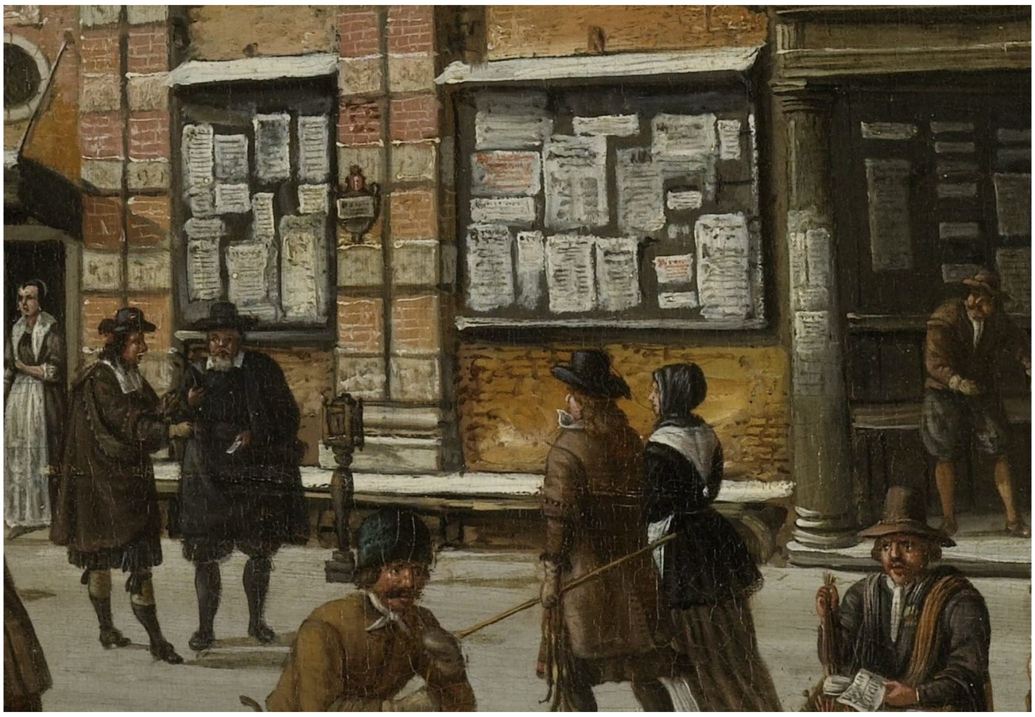


Figure 1: Detail from The Paalhuis and the New Bridge (Amsterdam) during the winter, by Jan Abrahamsz. Beerstraten, c. 1640 – 1666. Oil Painting, 84 × 100 cm. Source: `http://hdl.handle.net/10934/RM0001.COLLECT.5966`

The affixation of ordinances, or placards, to known places made them official, for if a rule remained unknown to the public, it could (and would) not be obeyed. The federation-states (Dutch: *gewestelijke staten* or 'provincial' estates[1]) considered it to be essential to also print a selection of their agreed-upon texts in books of ordinances (Dut. *plakkaatboeken*). These volumes formed, e.g. a source for lawyers as a reference work, but cannot be considered a complete overview. In most cases, they merely provide an indication of what government officials deemed essential rules. These folio books were much more manageable to handle than the original offprints that could size around

[1] The English translation *province* for the Dutch word *gewest* can be misleading, as it has the connotation of being subordinate to another entity – while the Low Countries' provinces were basically federation-states which is why we use this latter term.

30x40 cm, as the used font is much smaller too.

1.1 Hypothesis

Both the Dutch Republic and the Habsburg Netherlands were federations of autonomous states. In the Republic, the Estates-General held sovereign powers, and in each of the federation-states, the estates held the highest power. The Republic's Stadtholders were officially civil servants. The Habsburg Netherlands differed from the Republic as they had a sovereign prince (the King of Spain), though the federation-states did have a certain amount of freedom. They had to verify that new rules did not jeopardise traditions and customs. However, when one looks in many history books, the early modern European-scene is depicted as a conflict among the noble dynasties; in other words, there is a strong focus on monarchies.

This focus results in poorly studied political-institutional constellations of multi-layered republics - the Dutch Republic and Switzerland alike. We know too little to say something concrete about the rule of federation-states. While Belgium has a long tradition of republishing the rules through the Royal Commission for the Publication of Ancient Laws and Ordinances[2] (since 1846), the Netherlands do not hold such an institute. This knowledge-gap resulted in a study between Holland and Flanders – two federation-states that are trade-oriented – indicating that the differences in the Republic's and Habsburg's legislation were not that significant (Romein, 2019). Hence, the following hypothesis arose:

> Early Modern European states struggled for survival, making it impossible to 'reinvent the wheel' each time a problem arose. Hence, it was of tremendous importance to copy, adapt and implement normative rules (often understood as legislation) that were already proven successful elsewhere.

In order to be able to study this hypothesis, a massive amount of data needs to be generated, categorised and analysed. When that data is available, it will be possible to create a topical subset which will then allow textual comparisons. The first Digital Humanities building-block in this puzzle became the KB Researcher-in-residence[3] project *Entangled Histories*. In this article, we present research informed by this hypothesis as a preliminary analysis; however, a more rigorous analysis will be done in the future. In this article, we explain the process of datafication of the Early Modern Ordinances that consisted of text recognition, segmentation and automated topic classification of the sources. We elaborate on the challenges and prospects of this (type of) research.

1.2 Digitisation of Books of Ordinances

The digitisation of books of ordinances may not have been a priority of libraries, as most of these books are reasonably readable by researchers. However, this close reading requires reliance on the provided indexes - if any is available. The number of pages per book in our set ranges from 34 to 1921 pages and a lot of normative rules are contained in one volume, making it a challenge to read all and everything. Linguistic challenges such as spelling variation and the way of referencing to specific problems pose another challenge.

[2] See https://justitie.belgium.be/nl/informatie/bibliotheek/koninklijke_commissie_uitgave_belgische_oude_wetten_en_verordeningen

[3] See https://www.kb.nl/en/organisation/research-expertise/researcher-in-residence

For example, the word 'gipsy' (Dut. *zigeuner* or Fr. *manouche*) as a reference to the Roma or Sinti people is not in the texts, but the words *heiden* or *heidens* (Eng. *infidels*) are. As language - and thus, references - change over time, a book from the 1600s could have chosen a specific word in the index, whereas a book from the 1700s could have chosen another word - which complicates (automatic) searches. Full-text searches thus require complete access to the sources.

To be able to start the datafication-process, an overview of the available books of ordinances is required, which did not exist. Hence, as a by-product of *Entangled Histories*, we have published a list of 108 digitised books of ordinances.[4] The list is presumably incomplete, primarily since it is unknown which books of ordinances were printed. Hence, this should also serve as an invitation to inform us about excluded books.

The available digitised books can be distinguished into various groups, not just per publisher or per federation-state. Eighty-eight of these books were printed in a Roman-type font, twenty in a Dutch Gothic font - which differs from the German Gothic font (*Fraktur*). The language in the books varies among the regions, but the main languages are Dutch (67), French (26), Latin (1) and a mix of those (14). The total amount of pages is approximately 75,000, resulting in an estimate of 550 million characters to process. All books have been published between 1532 and 1789, within the seventeen federation-states of the Habsburg Netherlands and the Dutch Republic.[5]

1.3 Datafication

In this paper, we describe the datafication of the collected books of ordinances. We discern three phases in the datafication: text recognition, segmentation and categorisation, which provide the structure of this document.

We used Transkribus for text-recognition; which yielded good results for the entire corpus. Due to time limitations, we have chosen to work with a Proof of Concept for the next two phases. For this purpose we selected the *Groot Gelders Placaet-boeck, Volume 2*. The first author had previously made a list of titles with topic annotations for this volume, that could serve as a basis for our experiments. We experimented with several tools for segmentation, and applied Annif for the categorisation.

2 Text Recognition

108 books have been digitised either by the library's initiatives to digitise books (University Utrecht, Bodleian Library) or through the Google Books project[6]. These books have been processed for Optical Character Recognition (OCR) by a version of ABBYY FineReader. In the OCR-technology "[...] scanned images of printed text are converted into machine-encoded text, generally by comparing individual characters with existing templates"(Muehlberger et al., 2019, p. 955). Manual inspection of the OCR-output for some of the books in our study revealed they were completely incomprehensible. Unfortunately, the older the books are, the more problematic OCR is – especially when printed in Gothic script. Even the Roman-type font is a challenge with the long s (ſ) a character that resembles the f to a great extent.

[4] See `https://lab.kb.nl/dataset/entangled-histories-ordinances-low-countries` and archived in Zenodo: `https://doi.org/10.5281/zenodo.3567844`.

[5] It thus excludes the Prince-Bishopric of Liège and the principality of Stavelot, which were not

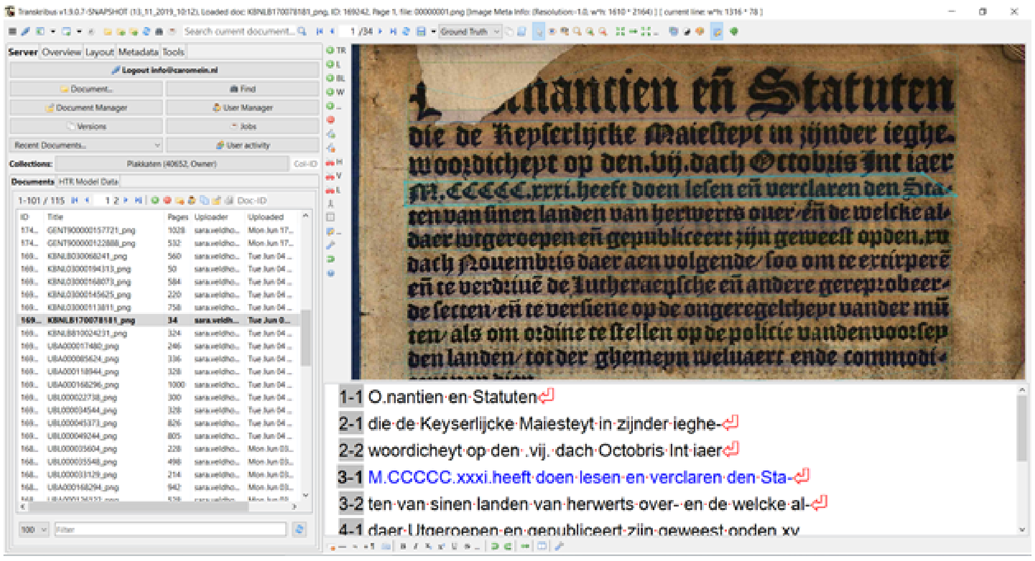

1-1 O.nantien en Statuten
2-1 die de Keyserlijcke Maiesteyt in zijnder ieghe-
2-2 woordicheyt op den .vij. dach Octobris Int iaer
3-1 M.CCCCC.xxxi.heeft doen lesen en verclaren den Sta-
3-2 ten van sinen landen van herwerts over- en de welcke al-
4-1 daer Utgeroepen en gepubliceert zijn geweest opden xv

Figure 2: Screenshot of Transkribus (v. 1.9.0.7) - showing a 16th century Dutch Gothic ordinance and the transcription.

2.1 Method: Automatic Text Recognition (ATR)

Properly training OCR tools such as Kraken, Tesseract or, indeed, ABBYY could lead to more recognition of gothic script (Tafti et al., 2016)[7]. However, we wanted to test another potential method: Handwritten Text Recognition (HTR). Within the Recognition and Enrichment of Archival Documents-project (READ) Transkribus applies this HTR-method.

Given the complexity of handwriting a combination of techniques is used: advanced pattern recognition is combined with artificial intelligence, and recurrent neural networks. These three are employed to recognise various hands after training (Muehlberger et al., 2019). This technique can also be applied to complex printed texts, such as Sanskrit, Cyrillic, or, indeed, early modern printed texts from Western Europe, such as Gothic (see Figure 3). As the technique is used for handwriting as well as print, the term Automatic Text Recognition (ATR) is applied. This tool has a graphic user interface, so it is a tremendous asset for the traditionally trained humanities-scholars allowing a transformation of sources into searchable data.

When applying ATR to printed texts, the user trains the computer to regard the characters as 'impeccable handwriting'. With that in mind, we expected to be able to recognise texts with a Character Error Rate (CER) of less than 5%.

The texts were uploaded into Transkribus in the PNG-format[8]. Then, the follow-

part of these entities.

[6] See https://books.google.nl/

[7] OCR is being developed further and other interesting tests could be made - but were not included in this project:, e.g. Konstantin Baierer, Rui Dong, and Clemens Neudecker. 2019. Okralact - a multi-engine Open Source OCR training system. In Proceedings of the 5th International Workshop on Historical Document Imaging and Processing (HIP '19). Association for Computing Machinery, New York, NY, USA, 25–30, DOI:https://doi.org/10.1145/3352631.3352638; Reul, C.; Christ, D.; Hartelt, A.; Balbach, N.; Wehner, M.; Springmann, U.; Wick, C.; Grundig, C.; Büttner, A.; Puppe, F. 'OCR4all—An Open-Source Tool Providing a (Semi-)Automatic OCR Workflow for Historical Printings'. Appl. Sci. 2019, 9, 4853.https://doi.org/10.3390/app9224853; or at https://ocr-d.de/.

[8] Transkribus allows several formats, including JPG, TIFF, PNG, PDF. However, it does not allow JPEG2000, the much compressed format in which most Google Books are saved. We therefore converted

ing steps were taken: (1) the automatic layout analysis, and (2) the partial manual transcription of the books to develop a Ground Truth to develop models in order to transcribe the rest of the texts automatically. It entailed manual transcription of a minimum of 50-75 pages per font-type/ per period.

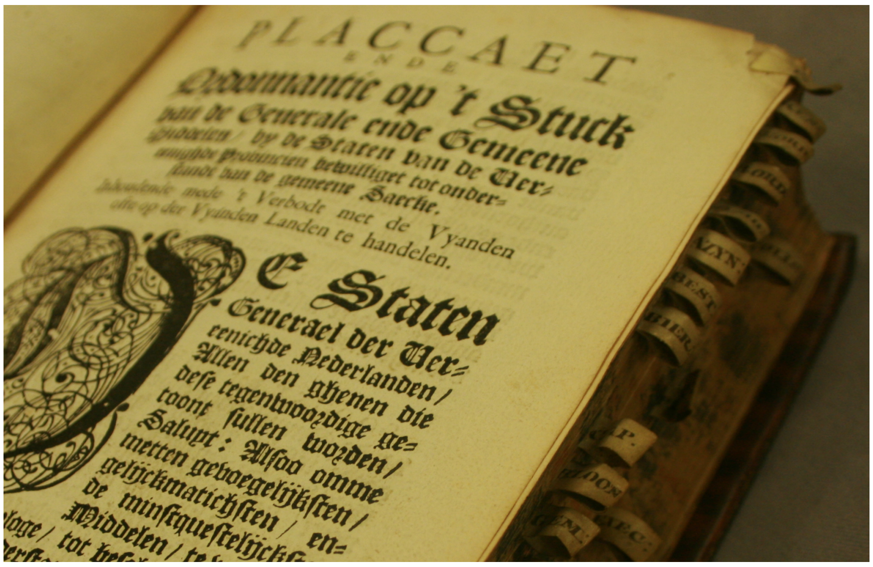


Figure 3: A Gothic printed book of Ordinances (KB National Library of the Netherlands).

2.2 Results: ATR

Within *Entangled Histories*, we chose to combine the material into groups related to the font and language. Hence, it has resulted in three self-created models: *Gothic_Dutch_Print*; *French_18thC_Print*; and *Romantype_Dutch_Print*.[9] For our Latin book from Artois, we applied the publicly available Latin model Noscemus GM v1 created by Stefan Zathammer.[10]

Each of these models has been created with the use of both a train and a test set. The test set was used to test the model and predict its ability to work for unseen material. It is crucial to prevent the overfitting of a model for a specific type of text. As a rule of thumb deep-learning-expert Gundram Leifert (CITlab University of Rostock/ Planet AI GmbH), who develops the HTR-component of Transkribus, advices to use a minimum of 1000 lines of text of which 10% is entered as validation.

The results as presented in Table 1, with CER's of 1.71% (Dutch_Gothic_Print), 0.65% (French_18thC_Print) and 1.17% (Dutch_Romantype_Print), exceeded our expectations as well as goals for the Entangled Histories-project tremendously. With those excellent CER-results, the datafication of the original source-material - the books of ordinances -

them to PNG using imagemagick (`https://imagemagick.org/`).

[9] The option to train models is not standard in the GUI, it needs to be requested by email (email@transkribus.eu). HTR+ is an additional feature, allowing the Recurrent Neural Networks to run more than the standard 40 epochs (in HTR) to 200 epochs (standard), although it can be raised to a manually altered number of epochs of over 1200.

[10] The model Noscemus GM v1 comprises 170658 words and 27296 lines, it shows a CER of 0.87% on the training set and 0.92% on the test set. This HTR+-model is tailored towards transcribing (Neo-)Latin texts set in Antigua-based typefaces, but it also, to a certain degree, able to handle Greek words and words set in (German) Fraktur.

Table 1: Results per created model (CER).

Model name [ID]	Training (CER)	Test (CER)	# Words (training)	# Lines (training)
Dutch_Gothic_Print [Model ID18944]	0.22%	1.71%	51143	7143
French_18thC_printed [Model ID19166]	0.33%	0.65%	38487	3883
Romantype_Dutch_Print [Model ID19423]	1.26%	1.17%	88105	13013

is even much better than expected. In other words, the texts will be well human- and machine-readable.

3 Segmentation

How can the computer segment a text? The human eye can easily spot different parts on a page, such as headers, footers, marginalia, titles and paragraphs in contrast to a computer. Within *Entangled Histories*, several tools were explored: *ABBYY FineReader v.11*, *P2PaLA*, *NLE Document Understanding* and a rule-based approach we created ad hoc as a backup. Initially, the expectation was that assigning layout structures within Transkribus would be regarded as text-enrichment, but it turned out to be the first step in the analysis process. Although we tested several segmentation-tools within *Entangled Histories*, we could not rely on them as they became available too late or have not yet reached their full potential. However, P2PaLA and NLE Document Understanding are very promising tools-in-development. We therefore created a tool for segmenting the proof of concept ad hoc.

3.1 Method

Several segmentation options were investigated on various volumes within the collection.

ABBYY FineReader v.11

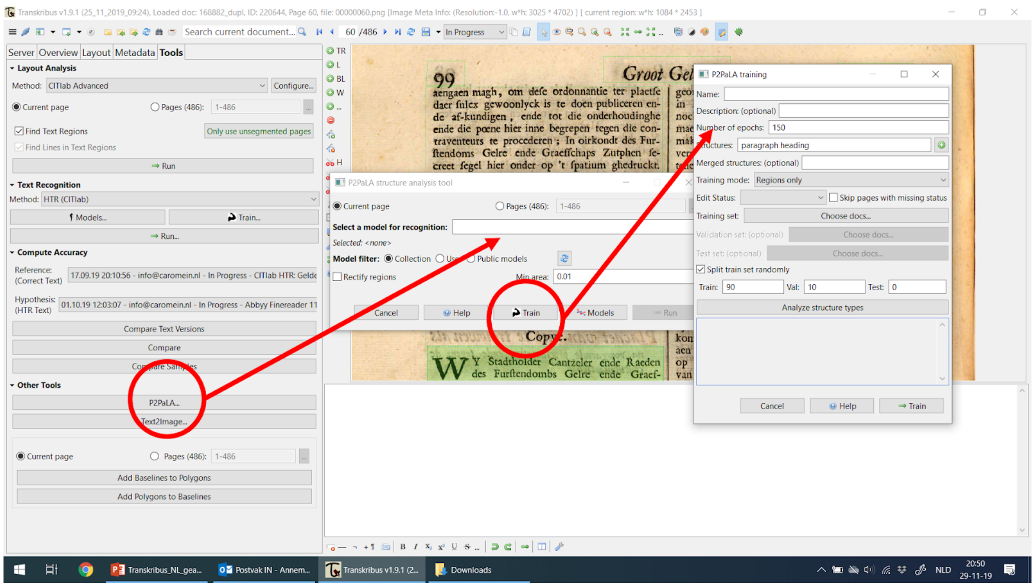
As an OCR-engine, built into Transkribus, we initially used ABBYY to recognise columns, whereas the Transkribus automatic layout analysis (LA) could not recognise them properly.[11] Interestingly, ABBYY also adds information regarding the size of fonts to the XML-data that is provided. Which - at least for printed texts – could be used to discern different parameters such as e.g. columns, font-size. However, this is not foolproof as ABBYY regularly fails or adds information where it should not.

P2PaLA (Alpha-version)

Lorenzo Quirós Díaz is developing Page to Page Layout Analysis (P2PaLA) at the UPVLC (Universidad Politécnica de Valencia), a READ-COOP within the research-centre: Pattern Recognition and Human Language Technology (PRHLT).[12] The idea(l)

[11] See `https://www.abbyy.com/media/10433/what-is-new-in-finereader-engine-en.pdf`
[12] See `https://www.prhlt.upv.es/wp/`

is to train several pages by labelling the text regions (without baselines) and telling the computer what is so special about the fields (Quirós, 2018). The models operate pixel-based, and (text) regions could be any shape. This flexibility comes with a price: many data points are needed per region type. At the moment of testing, we were advised to annotate about 200 pages per (type of) book, with at most five different region types to be tagged.


Transkribus v1.9.1 (25_11_2019_09:24), Loaded doc: 168882_dupl, ID: 220644, Page 60, file: 00000060.png [Image Meta Info: (Resolution:-1.0, w*h: 3025 * 4702)] [current region: w*h: 1084 * 2453]

Layout Analysis
Method: CITlab Advanced
Current page / Pages (486): 1-486
Find Text Regions ✓ — Only use unsegmented pages
Find Lines in Text Regions ✓

Text Recognition
Method: HTR (CITlab)

Compute Accuracy
Reference: (Correct Text) 17.09.19 20:10:56 - info@caromein.nl - In Progress - CITlab HTR: Gelde
Hypothesis: (HTR Text) 01.10.19 12:03:07 - info@caromein.nl - In Progress - Abbyy Finereader 11

P2PaLA structure analysis tool
Current page / Pages (486): 1-486
Select a model for recognition:
Selected: <none>
Model filter: Collection, User, Public models
Rectify regions — Min area: 0.01

P2PaLA training
Name:
Description: (optional)
Number of epochs: 150
Structures: paragraph heading
Merged structures: (optional)
Training mode: Regions only
Edit Status: — Skip pages with missing status
Training set: Choose docs...
Validation set: (optional) Choose docs...
Test set: (optional) Choose docs...
Split train set randomly ✓
Train: 90 Val: 10 Test: 0

Server
Overview
Layout
Metadata
Tools
Configure...
Run
Models...
Train...
Run...
Compare Text Versions
Compare
Compare Samples
Other Tools
P2PaLA...
Text2Image...
Add Baselines to Polygons
Add Polygons to Baselines
Cancel
Help
Train
Models
Run
Analyze structure types
99
Groot Gel
aengaen magh, om deſe ordonnantie ter plaetſe
daer ſulex gewoonlyck is te doen publiceren en-
de af-kundigen, ende tot die onderhoudinghe
ende die pœne hier inne begrepen tegen die con-
traventeurs te procederen; In oirkondt des Fur-
ſtendoms Gelre ende Graeffchaps Zutphen ſe-
creet ſegel hier onder op 't ſpatium ghedruckt.
Copy.
WY Stadthoulder Cantzeler ende Raeden
des Furſtendombs Gelre ende Graef-
In Progress
Search current document...
60 /486
Transkribus_NL_gea...
Postvak IN - Annem...
Transkribus v1.9.1 (2...
Downloads
NLD
20:50
29-11-19

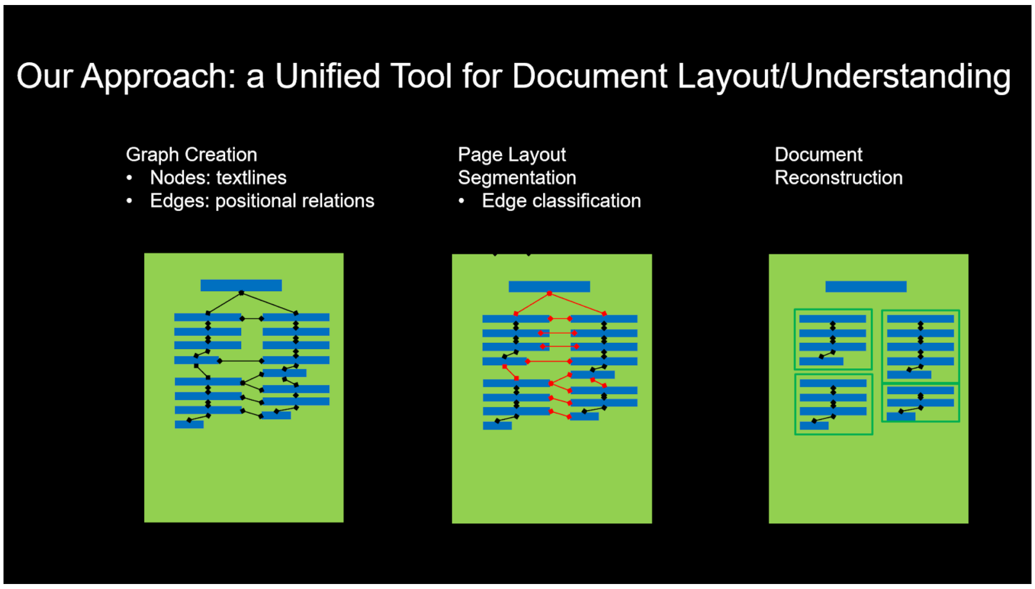
Figure 4: Screenshot of Transkribus (v. 1.9.1) - showing the training-screen for P2PaLA. In the right pop-up screen, several 'structures' (fourth field) can be selected to train an LA-model.

At the moment that we tested P2PaLA, it still required external involvement from Innsbruck to set up the training and implement the trained model into the P2PaLA-module. As of December 2019, a selected group of alpha-users can train their own P2PaLA-models within Transkribus and use this straight-away (as shown in Figure 4).

NLE Document Understanding

Like P2PaLA, NLE Document Understanding is a tool under development. NLE stands for Naver Labs Europe, which is one of the READ-COOP partners to develop tools to process texts better. At this point, NLE Document Understanding still requires external help to process the analysis, but it is expected to be incorporated into Transkribus in 2020. Using Artificial Intelligence, the page-layout is processed into nodes and edges and consequently classified in order to reconstruct the role and position of the text within the document (Clinchant et al., 2018, Prasad et al., 2019) (Figure 5). Here a crucial element, addressed by Koolen and Hoekstra (2019) at the DHBenelux 2019 conference, is obeyed: text was not placed at a specific spot by accident, there was a deliberate thought behind it.

Naver Labs' approach can be applied to tables as well as other document structures. Although very promising, the last-minute availability of this approach prevented us from exploring the options further.



Our Approach: a Unified Tool for Document Layout/Understanding

Graph Creation
- Nodes: textlines
- Edges: positional relations

Page Layout Segmentation
- Edge classification

Document Reconstruction

Figure 5: Slide Naver Labs Europe DevView 2019 (booth presentation) *READ: Recognition and Enrichment of Archival Documents. Digital preservation and discovery of the past* (slide 11/23). By Jean-Luc Meunier and Hervé Dejean.

Rule-based approach

Using the *Groot Gelders Placaet-boeck, volume 2* as the book to get a proof of concept on had one significant benefit: the 470 laws in this volume had been manually annotated with metadata including the titles and an indication on which page the title could be found. Using the information on font size from ABBYY whenever available, together with keyword matching on common title words, we developed a script[13] to trace the titles - and thus paragraphs - within the document. Although the wider applicability of this approach is quite limited, it enabled us to pursue our proof of concept.

3.2 Results

The layout of original documents secures much information. Not being able to recognise this structure with a computer and having to find means to re-implement the structure afterwards is a waste of energy, especially when one needs to reprocess HTR-models after applying lay-out analysis. When the returned results are adequate though, it will be worth the effort.

P2PaLA

We provided two books with structural tags indicating left/right paragraph, heading, header, page number and marginalia to be tested within P2PaLA. For each of the books, between 150 and 200 pages were marked. The results were ambiguous: one trained model gave ambiguous results. As this tool is still much in development, not much can be said about the results. This inconclusiveness - in the pre-alpha phase - left us no choice but to abandon this route for now.

[13] The code (XSLT transformation sheets and a Python notebook) for this rule-based tool can be found on github: https://github.com/KBNLresearch/EntangledHistories

NLE Document Understanding

The initial training on a handful of pages through NLE Document Understanding resulted in - at that moment - an accuracy of 85% correctly performed layout analysis. They claim to be able to reach a 95% correct performed layout analysis: providing that a training set of pages - including ATR-transcriptions - is representative for the document's structure.[14]

Rule-based

Based on a combination of typographic information derived from ABBYY and matching common title-words, we were able to recognise titles with 95% accuracy. We were able to segment the book into individual laws under the admissible hypothesis that all text following one title until the next title belongs to the same law.

4 Categorisation

Classifying documents as belonging to topics generally improves searchability. Moreover, topics or categories can inform one about the relations between texts from different books (provinces) without requiring close reading. As such, automatic categorisation could help fine-tuning a selection which would in turn allow the primary hypothesis to be answered. For now, we applied it to the single book in our proof of concept. We manually annotated the laws with topics from a controlled vocabulary and used the annotations to train an automatic subject indexing tool called Annif.

4.1 Method

We applied a controlled subject vocabulary, in which the norms were labelled with subjects from a categorisation created by the German Max-Planck-Institute for European Legal History (MPIeR). The same categories have been applied internationally, in over 15 early modern European states. It was developed in the projects Repertorium der Policeyordnungen[15], and Gute Policey und Policeywissenschaft[16] ran by Karl Härter and Michael Stolleis (Kotkas, 2014; Stolleis et al., 1996).

Within the MPIeR-project a four-level deep hierarchical categorisation considering police ordinances (public law) was designed. The books of ordinances also contain international laws that are out of the scope of these categories. For this reason, we added another level (level 1) to distinguish 'Police Legislation' and 'International Law'. In Figure 6, the five categories (at level 2) in Police legislation are displayed, together with subcategories that occur in our dataset. For levels 3-5, the number of available categories are 25, 163 and 1584 respectively. Note that the deepest level (level 5), is open for adding extra terms.

[14] Naver Labs Europe DevView 2019 (booth presentation) READ: Recognition and Enrichment of Archival Documents. Digital preservation and discovery of the past (slide 15 and 16/23). By Jean-Luc Meunier and Hervé Dejean. Data from the Passau archives and personal communications.

[15] See https://www.rg.mpg.de/forschungsprojekt/repertorium-der-policeyordnungen?c=2124983

[16] See https://www.rg.mpg.de/1928092/gute-policey-administrative-law-and-the-science-of-public-affairs

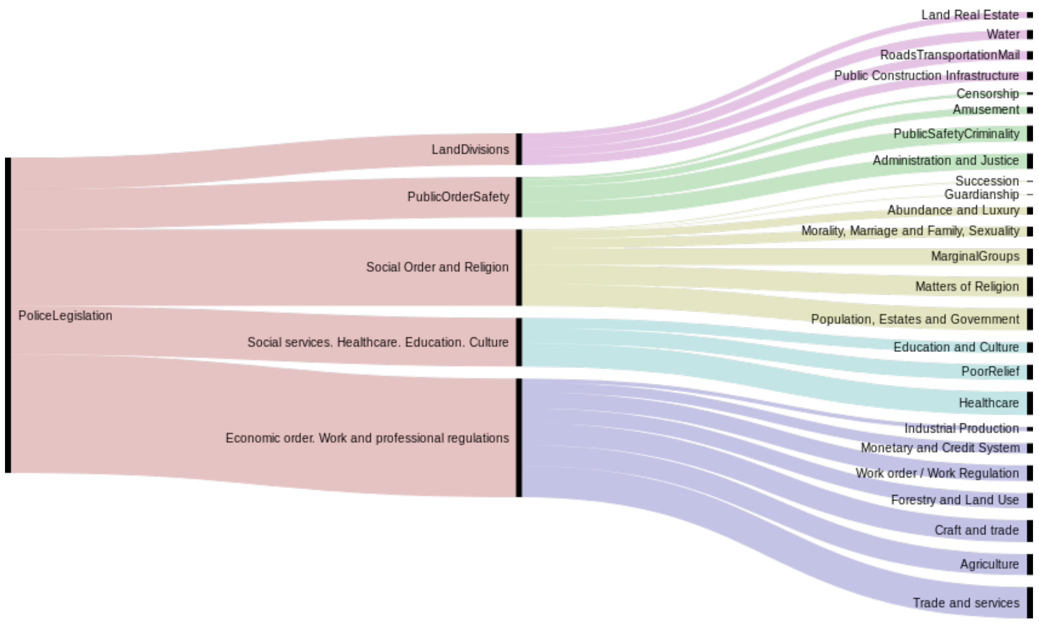


Figure 6: The first three levels for the top category 'Police legislation'. Bar width indicates the number of texts in each category. Through: Alluvial Diagram - `https://app.rawgraphs.io/`

Annotations

Since we are dealing with normative texts (legislation), the texts are relatively unambiguous compared to many other text genres. Manual annotations were added through close-reading and selecting the appropriate topic categories from the available list. Still, a single text can concern several (up to 10) topics at once. On average, each law was annotated with 3.3 categories, as detailed as possible: 69% and 28% of the annotations concerned categories at level 5 and 4, respectively. The topics are quite distinguishable until at least level three. For example, it is quite apparent to distinguish between several economic professions, or between primary school and university, or indeed, whether a rule applies to marriage or adultery.

Annif

Annif is a toolkit for automated subject indexing, developed at the Finnish National Library.[17] From existing metadata with subject headings from a controlled vocabulary, it can learn how to assign those headings to new, unseen data (Suominen, 2019). The tool comes with a variety of back-ends, ranging from lexical methods to vector-space models. It also offers ensemble learning, allowing one to combine the strengths of trained models from different set-ups.

In our experiments, we focused on TF-IDF first in order to see whether any reasonable categorisation could be found, as it is an accessible back-end that can be used without much adjustment. The terms (words) in every document are weighted by their frequency of occurrence (TF), and compensated by the inverse frequency in the entire corpus (IDF). The term frequencies in new documents are compared to those in existing documents, for which the subjects are known. For this, Annif uses the implementation in Gensim (Řehůřek and Sojka, 2010).

[17] Project description and link to source code can be found at `http://annif.org/`

Most back-ends, including TF-IDF, rely on stemming or lemmatization to unify inflections of content words. We used the Dutch snowball analyser as implemented in nltk[18], although ideally one may want to develop an analyser that is tailored for historical Dutch. Note that remaining character errors, as well as inherent spelling variation in historical texts, may influence both the stemming and the generalising capabilities of Annif models.

The hierarchical nature of the categories could be informative for the automatic categorisation. To allow the hierarchical structure to be imported into Annif, we transformed the vocabulary into a *Simple Knowledge Organization System-format* (SKOS) (Tennis and Sutton, 2008). The provisional SKOS-file is archived in Zenodo.[19] Unfortunately, most back-ends in Annif are currently unaware of hierarchy in the subject vocabulary, except for the Maui Server back-end.

4.2 Results

We were particularly interested in the performance of subject indexing at different levels (depths) of the subject vocabulary. Therefore, we created five versions of the dataset: one for each level, where the document would link to the hierarchical ancestor(s) of its assigned topic(s). In the subsequent analysis, level 1 indicates the distinction between international law vs. police legislation (i.e. public law) and the subsequent levels in the hierarchy of the MPIeR categories.

Due to the limited amount of data in our proof of concept-phase - a mere 470 laws - we ran all the experiments using a 10-fold cross-validation with a 90/10 train/test split. Due to time constraints, the only Annif back-end we have been able to experiment with so far was TF-IDF.

The HPC team of Ghent University (Belgium) has been so kind as to provide access to their infrastructure for running the categorisation experiments. Although Annif is not particularly heavy software, the different back-ends may be more demanding. As such, running experiments in an HPC environment could prove quite useful, especially when testing with more massive datasets.

Precision@1 vs. majority baseline

The majority baseline was determined per hierarchy level as the ratio of documents that were annotated with the most common category in that level. It was computed based on all data (no train/test split). In Figure 7, we present precision@1 to compare the model performance to the majority baseline. Precision@1 indicates the accuracy of the most probable category for every law, as proposed by the model. The test precision shows a lot of variances, which can be attributed to the limited amount of data (10% of 470 documents) on which the scores were based. Train precision is typically higher than test precision, indicating a lack of generalization that is probably due to the limited amount of data as well. For levels 1 and 2, the majority baseline was not even reached by the model. This is probably due to the imbalance in the data: there are relatively few examples to get informed about non-dominant categories. Deeper in the hierarchy, the distribution over topics is more even, and the majority baseline starts to fail, while the model performance remains stable.

[18] See `https://snowballstem.org/algorithms/dutch/stemmer.html`

[19] Find the SKOS ar `https://doi.org/10.5281/zenodo.3564586`

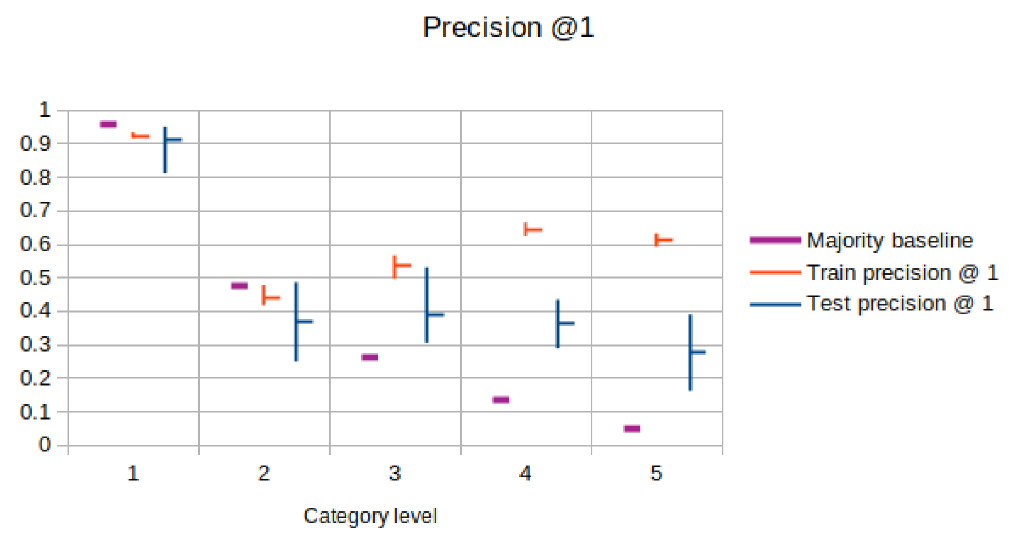


Figure 7: The horizontal axis indicates the hierarchical level of the subject indexing. The figure indicates majority baselines (purple), and precision@1 on the train (red) and test (blue) set. The horizontal bar indicates the average over 10-folds of the data; the vertical bar indicates spread.

Recall and precision with four predicted terms

More indicative of the actual performance of the model are recall and precision measured over all the model predictions. Precision indicates whether the terms suggested by the model are correct according to the manual annotation. Recall measures to what extent manually assigned terms are suggested by the model.

As usual, there is a trade-off between recall and precision: what counts as good performance also depends on the application. If one would use the assigned terms in an information retrieval set-up, high recall means that few relevant documents will be missed. In contrast, high precision will prevent irrelevant documents from showing up. In this stage, we optimised for F1: the harmonic mean of precision and recall. We determined the optimal results on all levels were obtained with a limit of 4 terms to be predicted using a threshold of 0.4, using the hyperparameter optimisation provided by Annif itself.

Figure 8 visualises the recall and precision. We present micro-averaged metrics because those are more robust against imbalanced data. The model is already able to suggest 40% of the relevant detailed terms (level 5) on the test set, which is quite impressive for this task with such a limited amount of data. Again, we see that performance on the train set exceeds that on the test set. This outcome indicates that adding more training data would likely boost performance.

5 Discussion: Facilitating Future Early Modern (Ordinances) Research

The hypothesis posed at the beginning of this article, regarding the cross-border influence of normative texts could not be tested thoroughly due to the limited time this project ran. The results did not go far enough actually to test the hypothesis yet. The hypothesis will be tested in future studies. However, manual verification was done -

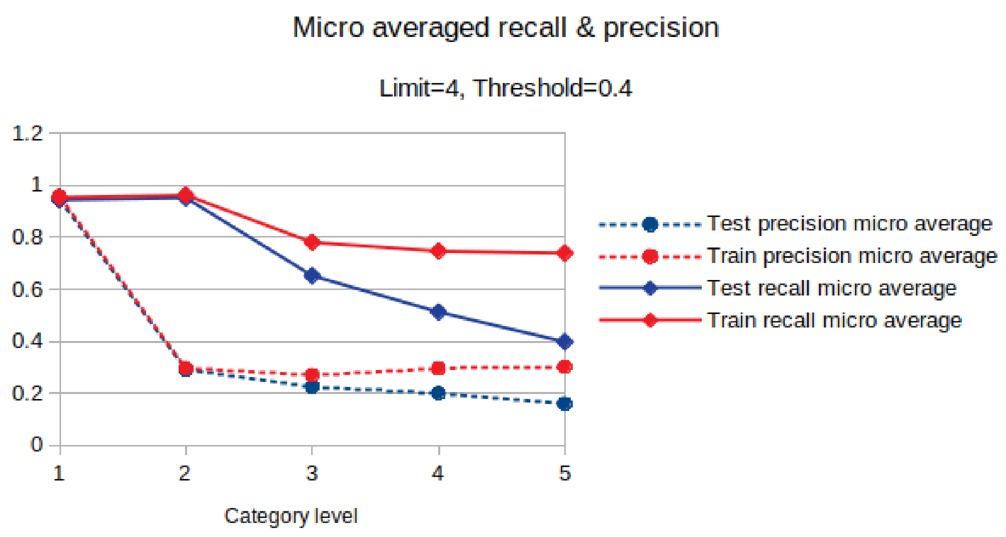


Figure 8: The horizontal axis indicates the hierarchical level of the subject indexing. The figure indicates average precision (dotted) and recall (solid) for the train (red) and test (blue) set.

due to the improved readability - through a full-text search within Transkribus itself. Such an approach obviously neglects the idea that categorisation looks at more context then just the searched key-word. It was already possible to establish a little proof of concept in those few cases that only a full-text search was used. For example, the case of a unification of the axle width to the Holland standard appears in Gelderland and Groningen. The topic of beggars and vagrants appears in multiple federation-states at the same time, though the formulation does differ (*deugniet* ('up to no good') vs. *beggar*). Such a quick search does not provide answers to the fullest extent the hypothesis envisions; hence, further research is needed.

Even if the transcripts are (nearly) perfect, early modern texts tend to contain a lot of spelling variations and dialects, as no 'standard spelling' existed. Furthermore, we know the transcription process is not flawless: contrary to digital-born texts, character errors exist that stem from the digitisation. These may prevent the unification of equivalent words, even after morphological processing. Moreover, remaining character errors may also negatively impact the performance of morphological tools. Within *Entangled Histories* we did not address these issues at this stage, nor did we apply a dedicated stemming algorithm for early modern Dutch. The extent to which this influences further processing, such as categorisation, would be an interesting direction for further research.

That a model is already able to suggest 40% of the relevant categories after being trained with only 90% of 470 texts is promising. Further research will tell us whether it is indeed possible to reach a much higher score by adding more texts and using more sophisticated back-ends. This testing will be done within *A Game of Thrones?!*-project (NWO Veni) at Huygens ING, which will take the *Entangled Histories* knowledge and results as a starting point and continues to work with them.[20] Using the case of Canton Berne (CH)[21] - previously studied in the MPIeR-project - Annif's ability to classify 5500 manually categorised texts will be tested, and the results now obtained

[20] See `https://www.huygens.knaw.nl/projecten/game-of-thrones/`.
[21] See `https://www.rg.mpg.de/2172198/volume007`

with 470 normative texts used in *Entangled Histories* verified. It should then become possible to automatically categorise sources from another Swiss Canton in the future. Furthermore, results from the federation-states Holland and Gelderland are a future basis to categorise Dutch ordinances of other federation-states automatically in the future.

In *A Game of Thrones*, other tools to segment texts will be considered.[22] The previously described tools will be reconsidered if improvements in technique have been made. In the case of Berne, matching with the existing list of ordinances from the MPIeR-project will be possible and will likely be helpful to validate the tools.

Other information that could be retrieved from these ordinances encompasses place names, dates, topics (categorisations), person names. In other words, performing Named Entity Recognition (NER) would be ideal for making the texts more searchable too. Once named entities have been recognised, one could visualise them on maps and timelines. These normative texts could be incorporated into Time Machine Projects[23] - which tend to leave the normative rules on the side.

Digital Historians - e.g. the Data for History Consortium[24] - are looking for ways of making geo-historical data interoperable in the semantic web. They suggest that this could be done through OntoME[25] which is designed for any object-oriented structured data model (based on CIDOC-CRM[26]), to make it easy to build, manage and align an ontology. Such an OntoME/Ontology for ordinances would be applicable on normative texts throughout Europe, providing the MPIeR-categorisation would be followed. Such an ontology would help solve language issues that occur in the current dataset, where French and Latin texts can be found in the Dutch collection (or vice versa). The OntoMe would help to structure the data in a machine-readable way, allowing to circumvent such challenges.

Early modern ordinances have long been left unattended, or at least research has not reached its full capacity. Datafication of these sources will bring more possibilities and will, in the longer run, enable us to see how cross-border influence (the entangled history) worked. One of the next steps is creating a Linked Data system to combine the available data. The map in Figure 9 shows - in light green - the research conducted at the MPIeR[27]; in dark green are other initiatives to inventorise the normative texts.[28] The multi-lingual SKOS will allow searches through different languages and, thus, across borders and through centuries. Hence, when this data does get connected and complemented with additional data, the possibilities to study the administrative norms of early modern (Western) Europe will almost become limitless.

[22] For example the rule-based, semi-automatic tool Layout Analysis tool Larex: https://github.com/OCR4all/LAREX

[23] See https://www.timemachine.eu/

[24] See http://dataforhistory.org/

[25] See http://ontome.dataforhistory.org/

[26] See http://www.cidoc-crm.org/

[27] See https://www.rg.mpg.de/forschungsprojekt/repertorium-der-policeyordnungen?c=2124983

[28] See e.g.: https://justitie.belgium.be/nl/informatie/bibliotheek/koninklijke_commissie_uitgave_belgische_oude_wetten_en_verordeningen; https://www.huygens.knaw.nl/projecten/resoluties-staten-generaal-1576-1796-de-oerbronnen-van-de-parlementaire-democratie/ [12-02-2020]; https://historischcentrumoverijssel.nl/digitalisering-historische-statenresoluties/ [12-02-2020]; https://www.huygens.knaw.nl/projecten/game-of-thrones/.

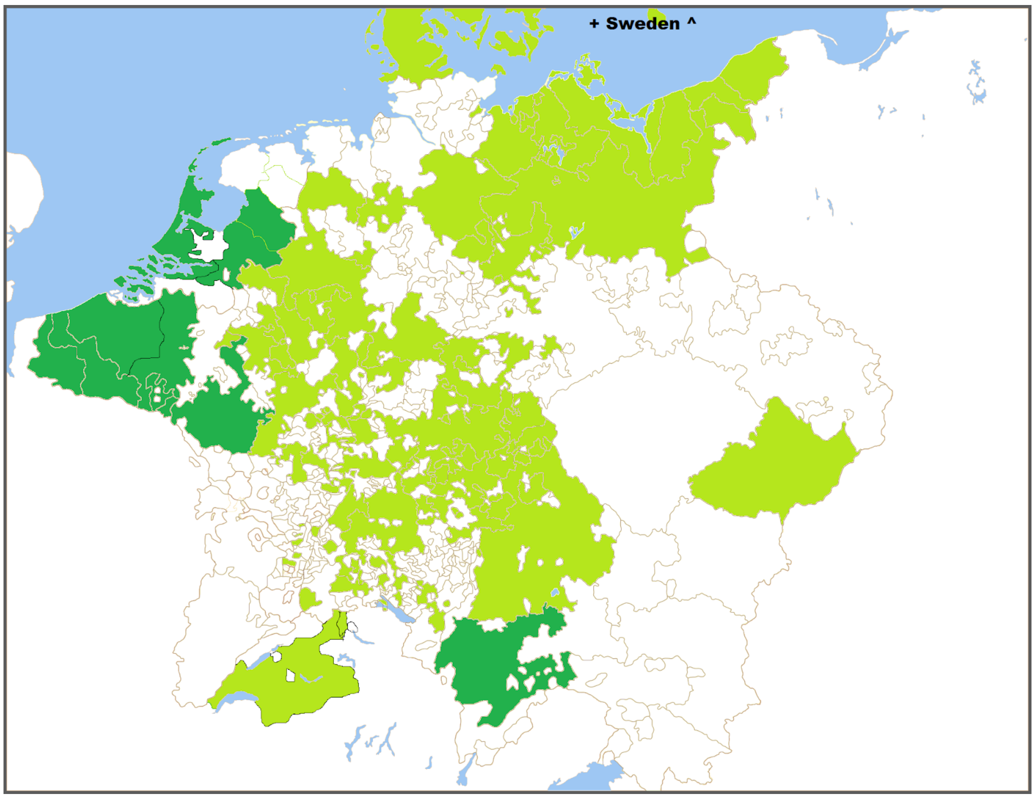


Figure 9: MPIeR Repertorium and other initiatives (dark green), period 1500-1800. Map: Blank map of the Holy Roman Empire in 1648, https://commons.wikimedia.org/wiki/File:Holy_Roman_Empire_1648_blank.png

6 Code and Data Availability

- The dataset used within Entangled Histories can be found at https://lab.kb.nl/dataset/entangled-histories-ordinances-low-countries and archived in Zenodo: https://doi.org/10.5281/zenodo.3567844.
- The code for the rule-based segmentation written by Sara Veldhoen is hosted at GitHub: https://github.com/KBNLresearch/EntangledHistories.
- The provisional SKOS used for the categorisation can be found at https://doi.org/10.5281/zenodo.3564586.

7 Acknowledgements

We would like to thank a number of people for their help and feedback throughout the project, which formed the basis of this paper: Martijn Kleppe, Lotte Wilms, Steven Claeyssens, Ronald Nijssen, Jeroen Vandommele, Rosemarie van der Veen-Oei, Mirjam Cuper and Ewald Pauwels; Julie M. Birkholz is thanked for proof-reading the article.

Annemieke would like to thank the KB National Library of the Netherlands for the hospitality and help with the project; in particular the team-members Sara Veldhoen and Michel de Gruijter. At Ghent University Annemieke would like to thank: René Vermeir, Dirk Heirbaut, Georges Martyn Thomas Donald Jacobs, Kevin Dekoster, and Klaas van Gelder as well as the members of the Institute for Early Modern History and the GhentCDH Research Lab for their continuous support.

8 Funding acknowledgement statement

This publication has been made possible through the KB National Library of the Netherlands Researcher-in-Residence-project. Each year - since 2014 - two researchers are appointed for six months to do research within the KB. The researcher can benefit from the knowledge and experience of our KB programmers and collection specialists, while the KB gathers valuable information to improve its services to digital humanities researchers.

This publication also benefited from the knowledge gained during the NWO Rubicon Grant 446-16-019 that Annemieke Romein received (2016-3) for her project *Law and Order: Low Countries? The relations between provincial and urban institutions in state-building in the Low Countries 1579-1701*.

References

Clinchant, S., H. Déjean, J. Meunier, E. M. Lang, and F. Kleber
2018. Comparing machine learning approaches for table recognition in historical register books. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, Pp. 133–138.

Der Weduwen, A.
2018. *Selling the republican ideal : state communication in the Dutch Golden Age*. PhD thesis, University of St Andrews. Not published.

Hoftijzer, P. G.
2015. *Europäische Geschichte Online : EGO : The Dutch Republic, Centre of the European Book Trade in the 17th Century*. Mainz: Leibniz-Inst. f. Europ. Geschichte.

Kleppe, M., S. Veldhoen, M. van der Waal-Gentenaar, B. den Oudsten, and D. Haagsma
2019. Exploration possibilities Automated Generation of Metadata. `https://doi.org/10.5281/zenodo.3375192`.

Koolen, M. and R. Hoekstra
2019. Reusing Existing Structures for Access to Large Historical Corpora. Presentation at DHBenelux 2019, Liège. `http://2019.dhbenelux.org/wp-content/uploads/sites/13/2019/08/DH_Benelux_2019_paper_10.pdf`.

Kotkas, T.
2014. *Royal police ordinances in early modern Sweden : the emergence of voluntaristic understanding of law*, The Northern world, 1569-1462 ; vol. 64. Leiden [etc.]: Brill.

Muehlberger, G., L. Seaward, M. Terras, S. Ares Oliveira, V. Bosch, M. Bryan, S. Colutto, H. Déjean, M. Diem, S. Fiel, B. Gatos, A. Greinoecker, T. Grüning, G. Hackl, V. Haukkovaara, G. Heyer, L. Hirvonen, T. Hodel, M. Jokinen, P. Kahle, M. Kallio, F. Kaplan, F. Kleber, R. Labahn, E. M. Lang, S. Laube, G. Leifert, G. Louloudis, R. McNicholl, J. Meunier, J. Michael, E. Mühlbauer, N. Philipp, I. Pratikakis, J. Puigcerver Pérez, H. Putz, G. Retsinas, V. Romero, R. Sablatnig, J. Sánchez, P. Schofield, G. Sfikas, C. Sieber, N. Stamatopoulos, T. Strauß, T. Terbul, A. Toselli, B. Ulreich, M. Villegas, E. Vidal, J. Walcher, Weidemann Max, H. Wurster, and K. Zagoris
2019. Transforming scholarship in the archives through handwritten text recognition: Transkribus as a case study. *Journal of Documentation*, 75(5):954–976.

Prasad, A., H. Déjean, and J. Meunier
2019. Versatile layout understanding via conjugate graph. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, Pp. 287–294.

Quirós, L.
2018. Multi-task handwritten document layout analysis. *CoRR*, abs/1806.08852.

Romein, C. A.
2019. De computer de wet laten herschrijven...?! Presentation KB Weetfabriek 23 September 2019, see https://www.youtube.com/watch?v=XZzL5j_sjkw. Archived on Zenodo: https://doi.org/10.5281/zenodo.3562881.

Stolleis, M., K. Härter, L. Schilling, and M.-P.-I. f. E. Rechtsgeschichte
1996. *Policey im Europa der frühen Neuzeit*, Ius commune (Klostermann).: Sonderhefte. V. Klostermann.

Suominen, O.
2019. Annif: DIY automated subject indexing using multiple algorithms. *LIBER Quarterly*, 29(1):1–25.

Tafti, A. P., A. Baghaie, M. Assefi, H. R. Arabnia, Z. Yu, and P. Peissig
2016. OCR as a Service: An Experimental Evaluation of Google Docs OCR, Tesseract, ABBYY FineReader, and Transym. In *Advances in Visual Computing*, G. Bebis, R. Boyle, B. Parvin, D. Koracin, F. Porikli, S. Skaff, A. Entezari, J. Min, D. Iwai, A. Sadagic, C. Scheidegger, and T. Isenberg, eds., Pp. 735–746, Cham. Springer International Publishing.

Tennis, J. T. and S. A. Sutton
2008. Extending the simple knowledge organization system for concept management in vocabulary development applications. *Journal of the American Society for Information Science and Technology*, 59(1):25–37.

Řehůřek, R. and P. Sojka
2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, Pp. 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

A-poetic Technology. #GraphPoem and the Social Function of Computational Performance

Chris Tanasescu (Margento)[1], Diana Inkpen[2], Vaibhav Kesarwani[2], and Prasadith Kirinde Gamaarachchige[2]

[1]UC Louvain
[2]University of Ottawa

Keywords: computational poetry, performance, natural language processing, graph theory applications.

The *Graph Poem* project, started at University of Ottawa in 2014 and continued meanwhile both there and at UCLouvain, has developed computational poetry classifiers that are deployed in representing poetry corpora as network graphs which are, in their turn, analyzed for graph theory-reliant features that will reflect back on the corpora and the poems thereof. This paper is part of a forthcoming cluster of publications that takes our focus beyond the strict affiliation of computational analysis, natural language processing (NLP), and graph-theory applications and into a wider digital humanities (DH) context. This complementing direction explores the poetry and poetics of DH and a possible "plus-poetics" as manifest in programs and performances like ours or David Jhave Johnston's *Big Data Poetry* (Tanasescu (2021)), the tightly-knit intercorrelation between digital writing, control, and "monstrous iconicity" in digital space and media (Tanasescu and Tanasescu (2021)), and the present writing aiming to foreground the social and community relevance and impact in digital space-based performances such as the Margento #GraphPoem EPoetry events presented at Digital Humanities Summer Institute (DHSI) in 2019 and 2020.[1]

One of the concerns in our previous NLP publications was to make a point in favor of poetry's relevance beyond the genre per se, in the wider framework of NLP and DH. For instance, the metaphor classifiers we developed trained on poetry and non-poetry data turned out to be better than the ones trained by other authors on just the latter or by us on just the former (Kesarwani et al. (2017)), which also helped with a deep-learning approach to metaphor (Tanasescu et al. (2018)), while important NLP instruments such as word embeddings trained on poetry corpora turned out to be better for any other text analysis purposes than other 'household name' ones (Tanasescu et al. (2018)).

[1] See https://bit.ly/2ASWDYl; the recording of the #GraphPoem @ DHSI 2020 performance can be watched here: https://bit.ly/2Nmk5j1; for the automated #GraphPoem tweets at DHSI 2019 and DHSI 2020 search for the @GraphPoem bot's profile page on Twitter.

In the present paper, we want to extend that argument regarding poetry's relevance 'beyond poetry' to the realm of the social and the political. In that respect, we find Jonas Andersson Schwarz's revisiting of the concept of *Umwelt* (in Uexkull's philosophy, cf. Schwarz (2018)) quite useful for our argument as it foregrounds nuanced notions of milieu and ecology that can correlate communities or social assemblies to various other kinds of ensembles. Digital space-based poetic corpora, for instance, are such ensembles, alongside the social ones, and can be better understood and worked on in contexts transgressing commonly accepted borders between the organic and the technical, as well as between the human and the machinic. In translating this notion to our world of artifacts, Schwarz draws on Johansson to highlight the fact that artefacts become ecological entities only as long as they are "attended to and used," thus becoming "as much an agent (co-agent) in the social ecology as is the organic human being" (Schwarz (2018), p. 66). We are particularly enticed by this fundamentally operational (or rather, as argued below, *performative)* and socially connecting nature of artifacts, especially in a framework whereby, in the footsteps of Alfred Gell, "the mediated environment prompts human self-understanding to take on *mental categorizations* that are isomorphic to this environment" (Schwarz (2018) 61, author's emphasis).

We take this ("human") self-understanding to be the poem's (self-)reading and (self-)performance as informed by its corpora and corporeal environment. Moreover, we see the corpus-based performance of the poem and the networked textualities thereof as potentially isomorphic to the societal connectivity and the radical approaches to community they are shaped by and/or feed back into. Our very concept of, and actual term, the *graph poem* (singular) refers to the multitude of poems in a corpus algorithmically analyzed and expanded as a network (i.e., mathematical 'graph') that together amount to, or rather asymptotically tend towards, a (locally) global encompassing poem. At the same time, every single node in such a network—every single poem indeed in the corpus—is informed and performed by that specific surrounding milieu of various (globally) local communities/subnetworks, or Umwelt(s).

We will involve three key agents in our notion of performative networked sociality of poetry in digital culture: humans, poems (as performative inscriptions in digital space)[2], and machines (as computationally implemented algorithms and artificial intelligence). While we do not believe that these agents are one and the same, we maintain that they deeply and intimately overlap in respects remarkably relevant to the (post)digital. And that overlapping is in our view made possible by the "isomorphism," in the quotation above, between "mental categorizations" and the mediated environment. Still, while those categorizations are for Schwarz—via Gell—instrumental in "human" "self-understanding," for us they are rather the very vehicle for shaping isomorphic networked milieus of humans/poems/machines. And in doing so, they shape the networked individuals across these interconnected milieus as well. It is in fact off of such isomorphisms that we base our notion of the potentially radical societal and community-oriented relevance of poetry in digital space and culture. Our umbrella term for such relevance is *data-commoning webformance,* which we will detail in a bit.

But before doing that, let us look more closely into the symmetrical bi-directional process we just alluded to, whereby environments and individuals are shaped and

[2] We are using the phrase "digital space" in Stephen Kennedy's acceptance of the term, the (ubiquitous/all-engulfing) space of chaos media that ontologically goes beyond the real-virtual binary and that is informed by a non-representational paradigm and by the "sonic economy" best describing digital culture (Kennedy (2015)).

shape each other simultaneously. We concur in this perspective with authors in the field of constructivist theories of cognition in which the crucial hypothesis is that "living systems are not primarily defined through the discrete qualities of their components, but through relations" (Schwarz (2018) 66). Our model is consistent with such theories in that we see poems—in digital media—as never in isolation, but always inscribed in the medium as performative and contextual. Their constitutive features are always informed by the (computational processing of the) other poems in the relevant corpus/ora, just as they are performed by the other writing operations involved in the inscription—and, again, computational processing—of those corpora in digital space and media. In terms of digital writing—as computational inscription and processing—the poem performs its environment while being itself performed by the latter, or, in the language of the above-mentioned theories, "the object comes to appear as if it generates an Umwelt" (Schwarz (2018)).

We would like to extend the scope of this model towards correlationist philosophy and explore the latter's previous translation to the subject of poetry—mainly in Brian Kim Stefans's *Word Toys. Poetry and Technics* (Stefans (2017))—and thus consider it as a possible basis for discussing poetry's potentially radical societal and community-oriented relevance. Stefans's poetics and references, remarkably rich and far too complex to be fully addressed in a discussion like the present one, will nevertheless provide more opportunities to explore than just the correlationist one. Yet for the latter already, the way in which Stefans draws on philosophers such as Quentin Meillassoux and writers like Vilém Flusser in, for instance, highlighting the notion that "only the correlation of the mind and object is what matters—neither can be understood without the other" (Stefans (2017), p. 1-2) will prove truly pertinent to the point we want to make.

We are (re)reading that latter statement from the dual angle of the isomorphisms above and the computational correlations informing our graph theory-based model. Our contention is that "mind-object" correlations, that is, connections made possible by means of (mutual) reading/processing/performance, refer to any (two or more) nodes in our networks. Poems inscribed and computationally processed in digital media are therefore understandable—they indeed can only exist actually—only through the correlations between one another. Every single poem is the mind while all the others are that mind's objects.

Further on, correlate sets of "mind categorizations" will ensure the propagation of the intra- or inter-corpus correlations into the social. Such processual and performative model involves manifold feedback circuits: the societal impact will reflect back on the graph poem's dataset(s), (re)configurations, and dynamics, which in turn will (help to) revisit, reshuffle, and at times reformulate the mathematics and algorithmics behind it, which in turn will impact the technology, platform, and web-based venue choices and/or constraints, and so on and so forth.

We are far from pioneering in trying to conceptualize poems as actors in a wider onto(techno)logical universe (or, rather, multiverse) shaping their own milieus. Although the actual societal values of that is different in his vision than ours, Brian Kim Stefans says something consistent on at least a couple of levels: "I'd like to imagine poems as autonomous entities that, like machines and living organisms, enact their own interactions with their milieus, perhaps each with its own 'will to power' and desire to reproduce, obtain sustenance, and evolve" (Stefans (2017), p. 2).While we are not attracted to the concept of the poem's autonomy and its being a "non-textual and even non-cultural object" (Stefans (2017), idem), we are indeed of the same mind in terms of a deep similarity between poems, on the one hand, and machines and "living organisms," on the other, especially in their interactive shaping of (and, naturally, being shaped by) their milieus.

Stefans draws in fact on major 20th century philosophers, perhaps most predominantly on Simondon and the latter's philosophy of technics, to spectacularly apply such approaches in poetry. He notably borrows Simondon's breakdown of technics into elements, individuals, and ensembles, where elements have no actual functional autonomy (e,g. wheels), individuals represent coherent and autonomous assemblages of elements (e.g. locomotives and cars), while ensembles link several objects (elements, individuals, etc.) into chains of production (cf. Stefans (2017), p. 61 et infra). Elements are characteristic of the whole pre-industrial era, the rise of industry coincides with the rise of individuals, and ensembles represent the technology of the "information age," the laboratory and, we would say most significantly, the computer network (idem).

While he is obviously not the first to explore the technical and, moreover, technological nature of poetry—major modernists such as, most popularly William Carlos Williams, but also Ezra Pound, have started a tradition that culminated with the rise of digital poetry—Stefans, while an outstanding practitioner of the latter, is the first to pursue this notion into developing a large-scale extensively theoretical and intensively illustrative poetics that refers to poetry in general; inclusive, that is, of both traditional 'page-based' as well as (post)digital (sub)genres. As already alluded, a work of such complexity and scope cannot possibly be properly discussed within the space and time of this writing, and that is why we will limit ourselves to highlighting some of the main concepts and particularly the ones relevant to our own topic and approach.

"I would like to suggest that poems, particularly the lyric (or short poem), 'succeed' to the level that they approach something like the technicity of a technical individual" (Stefans (2017), p. 69) states Stefans opting for the individual on Simondon's historical and typological scale. Poems are in his view therefore non-machinic coming-to-terms with the "presence of technical essences," and, again in the French philosopher's terms, the continuation of life "by means other than life" (Stefans (2017), idem). One may find striking this preference of the poet for the individual, but in fact, Stefans revisits the philosopher's classification and sees in the poem an ensemble "elevated ... to a technical individual" that, just like a machine, has parts performing functions of interaction with... *a milieu*, in ways similar to Pound's own "condensare" (cf. Stefans (2017), idem).

It is on this preference for the technical individual that incorporates features of the ensemble, particularly in interacting with the milieu, that Stefans bases his appropriation of Simondon's distinctions as adaptable to poetry. His own resulting innovative vision establishes, like above, tie-ins with certain foundational tenets of modern poetry and poetics. And as we will see below, this particular way of adapting such philosophical concepts to the subject will also allow Stefans to further tap into Simondon's notion of individuation and discover more valences relevant to, and useful for, poetry there as well. But we need first to outline his own vision on the milieu and its relevance to the social in as much as poetry is concerned.

Stefans argues therefore that, since, as Simondon elucidates, technical individuals have the highest degree of technicity, they actually present the evolution of machines towards what is "nearly organic" (63, author's emphasis). Consequently, their constitutive elements, the "infra-individual technical objects," (Stefans (2017), idem) have no associated milieu, as they simply work as parts of the individual "like the heart or liver in the body." While the constitutive elements have no milieu, the individual, the genuine invention, will characteristically require one, since its relationship to the latter are not mediated by any other technical entities. Translated for poetry, this means, very much in line with Stefans's poetics, that poems are self-contained technical objects with components ("elements") that interact only between themselves and that, as a whole, do not necessarily have a society around them. Neither the society of other poems (Stefans speaks little of corpora, and poems relate to each other mainly through an inherent generative diagram, when and if the latter is the same[3]), nor human society.

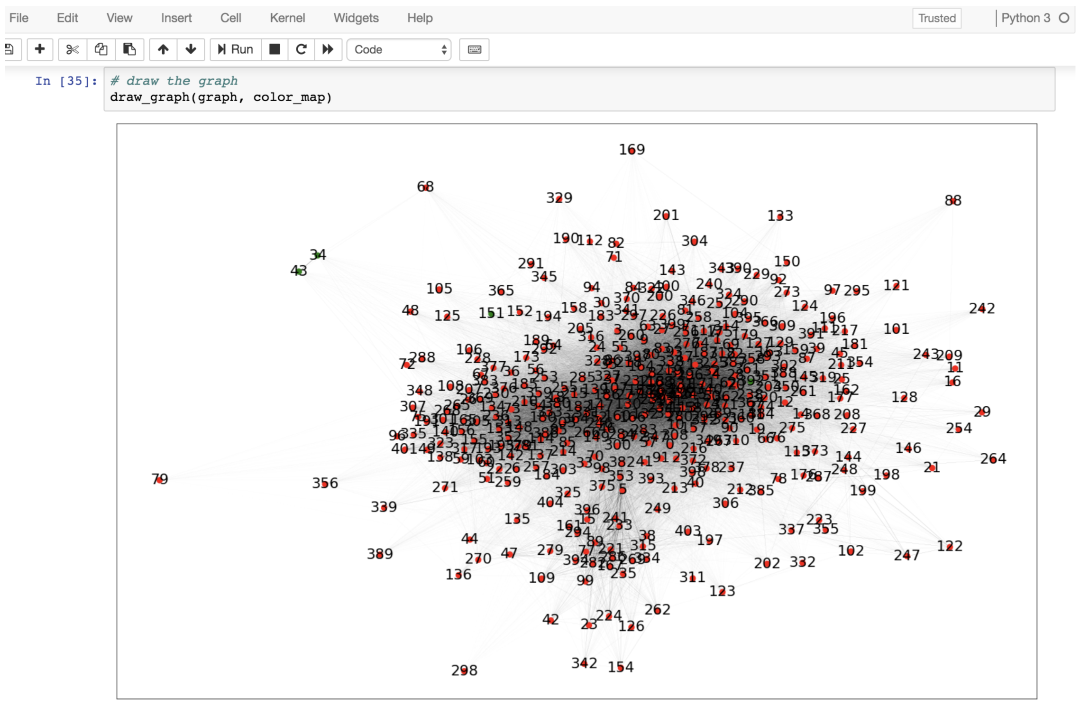
The latter aspect is in and of itself at the crux of Stefans's poetics. As he frontally announces from his introduction and already briefly mentioned here, to him poems are "non-textual and even non-cultural objects" (Stefans (2017), p. 2), a perspective that is totally consistent with his reading of Simondon's ideas on the development of technics as having its own unique track of evolution. "apart from (and certainly not dependent upon) social or economical developments, one that can literally outpace culture's ability to absorb these changes into art, philosophy, or behaviors..." (Stefans (2017), p. 60) Does this mean that he has no interest in the social dimension of poetry? That is not the case at all, yet his is—in this respect as well—a rather one-of-a-kind approach.

There are two features of the poem that have or can acquire a societal impact for Stefans. The first one refers to the fact that poems can work as *pharmakon* (Stiegler's concept based on Simondon's individuation) and are *evental* (Badiou's term for revolution as singularity creating new possibility from the void, and thus paving the way for the impossible). It is the former that is relevant to our discussion, so we will briefly outline the concept and then look critically into its possible interconnections with the computational poetry performance in our focus.

To Simondon (as revisited by Stefans) individuation is a person's growth into a singularity by tapping into their pre-individual grounds that can work as a, if not *the*, source of creativity (cf. Stefans (2017), idem). Interestingly enough, the social valence of that emerges as a form of therapy, social therapy consisting in the rapprochement between humans and technical objects, a notion that has been developed by Stiegler into his own well-known concept of *pharmakon*. It is the latter that Stefans appropriates and applies in his discussion of poetry, thus foregrounding the poem as the (rediscovered/recuperated) toy that can liberate us from the stinted, narrowly pre-defined, and alienating social relationships we are entrapped in. In doing so, pharmakon can help us reconnect with, or even regain, infinity as existential amplitude and unfettered sociality. While revisiting Anne Waldman's verse and using such reading keys, for instance, Stefans excavates from her verse poetic strategies to "establish the long-circuit of fidelity to the milieu" (p. 97) and thus illustrate Stiegler's pharmakon and the very definition of "infinite thought" at the same time. (98)

It is important to note that Stefans's mediation of Simondon's thought is itself an assertion of the poet's own poetics and consequential to the way in which he articulates the (non) sociality of poetry. His adaptation and development of the concept of pharmakon for poetry criticism is based on a definition of Simondon's individuation that comes with an impactful reduction. It is limited to "persons" (and their growth and becoming creative), which inevitably reduces (the) sociality (of poems) to human society, while in fact the philosopher conceived of individuation as applying "to

[3] Although this may of course remind one of New Criticism, Stefans's is a rather radical political and literary vision (inspired for example by Alain Badiou in ideology and the pataphysicians in literature) and more redolent of the Californian anarchist tradition rather than other more (or over) orthodox lineages.



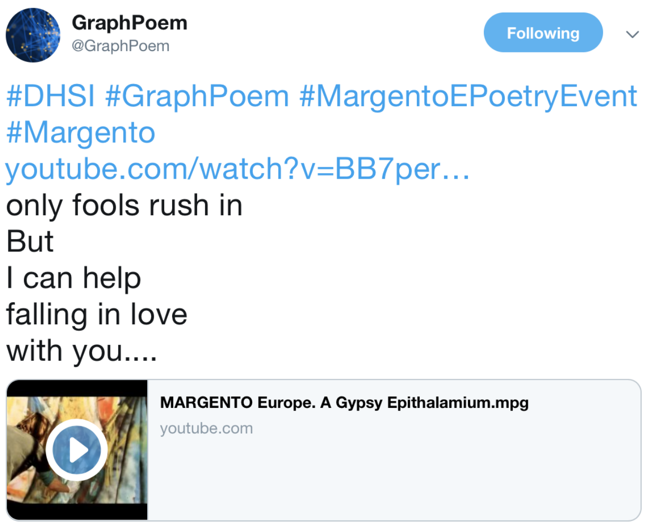
```
In [35]: # draw the graph
         draw_graph(graph, color_map)
```

Figure 1: #GraphPoem @ DHSI 2019 JupyterHub Notebook.

molecules, human beings, technical objects, and collective societies alike" (Schwarz (2018), p. 63).

Poems therefore, as technical objects (or rather, as Stefans strongly argues, technical individuals) can also know individuation, and they do so in, and by means of, interacting with their milieus. And, as stated earlier on, a model informed by environment-sensitive (mental) categorizations can ensure isomorphisms (or at least consistent and chartable correlations) between various kinds of milieus, in this particular case, networks of poems and online communities involved in data-intensive networked computational poetry performance. The Margento #GraphPoem EPoetry event at DHSI 2019 was announced as a performance and consisted of a JupyterHub Python notebook available on the University Victoria server only for the duration of the event (Figure 1)—authored by Chris Tanasescu—for collective live data collection, code running, and output visualization, and a bot (@GraphPoem)—programmed by Prasadith Buddhitha—that tweeted content outputted by the script on JupyterHub combined with text visualization and YouTube videos (Figure 2). The latter featured cross-artform work by Margento, the performance poetry band whose name has been transferred over time to an international collective of writers, artists, coders, and translators doing collaborative writing/art/performance, as well as to the team working on the DH project #GraphPoem. The participants in DHSI and the concurrent ADHO SIG DH Pedagogy Conference could access the JupyterHub script and run it on a corpus of poems assembled by us yet also available for them too to enlarge (with basically any text they found relevant, interesting, appealing, and/or simply random).

Although the event had been announced as a performance, the participants and other people in the audience were for quite some time confused and waiting for the 'gig' to start. Still, as the bot was intermittently tweeting and using a couple of relevant hashtags, more and more people started getting push notifications and at times nudging each other, "look, it actually already started, it's on Twitter, etc." As


GraphPoem
@GraphPoem
Following

#DHSI #GraphPoem #MargentoEPoetryEvent #Margento
youtube.com/watch?v=BB7per…
only fools rush in
But
I can help
falling in love
with you....

MARGENTO Europe. A Gypsy Epithalamium.mpg
youtube.com

Figure 2: The @GraphPoem bot tweeting at DHSI 2019.

the tweets came with visualizations and/or videos, soon participants, guests, and spectators starting following or watching the same or, most often, different things as simultaneously as part of the same event. These various mediations and temporalities and the interactions they spawned amounted to an enactment of the model on various levels and to a (number of) multilayer performance(s).

The term for such interactive community-oriented digital space-based performance as advanced in a previous publication is "commoning" (see Tanasescu (MARGENTO) (2016)). The term draws, on the one hand, on contemporary (post-Occupy) radical ('strike') art and/as performance, and, on the other, on recent radical thought (Hardt and Negri, for instance, and their description of the occupying multitude as a performance, cf. Tanasescu (MARGENTO) (2016) p. 12). Moreover, it also gestures towards non-essentialist approaches to community (Agamben's "coming community," Tanasescu (MARGENTO) (2016). pp. 20-2), and thus refers to (re)shaping/sharing/founding/finding community in/as communal enactment and/or collaborative performance (of the 'commons'). In articulating the latter aspect, we rely on recent advances in performance (and memory) studies and practice, particularly the work of Mechtild Widrich on "performative monuments" and reperformance (cf. Tanasescu and Tanasescu (2021)).

In our particular case—in as much as the event under discussion is concerned but also for #GraphPoem as the overarching project—the commoning is enacted in three major ways. First, by means of shared and collectively expanded data (in the form of txt file corpora as well as NLP and network generation and analysis algorithms in Python code), second, by live interactive coding script running at a 'commons' on JupyterHub, and third, the algorithmic 'communal occupation' of Twitter. We see all of these components and activities as fundamentally performative, particularly since digital space-based and all of them informed by digital writing (in its turn, essentially performative, see Tanasescu and Tanasescu (2021)).

Performance still, as already suggested, spills into quite a number of other aspects, media, and interrelations in potentially relevant societal ways. The manifold interactions of the performers/audience—between themselves and/or/by means of the various computational components, media, temporalities, and activities—represent to us a good opportunity to explore 'in action' the earlier on discussed isomorphisms between various entities such as humans, poems, and machines engaged in (milieu-driven/shaping) individuation in digital space and media.

We are particularly preoccupied with the individuation—and the inextricably intercorrelated Umwelt(s)—of poems. From the experience of events such as the one mentioned and the ongoing work with, and on, poems and text in digital media, a reality—with its attending poetics—emerges sensibly different from the one depicted by Stefans. A poem does not exist in digital media otherwise but as contextual inscription (enacted/performed always in relation to other texts and/or digital writing operations) and variable instantiating readings (read as instantiations of various contextually/operationally relevant features). In our particular case, including a poem in the DHSI event corpus already deployed a number of other digital writing operations related to the generation/instantiation and the format and location of the directory. On JupyterHub, on the other hand, the script kept 'reading' the new files *as part of the expanding corpus*, therefore as related to other (existing or potential) items in that directory. Furthermore, as the participants run the script, the latter will process the texts and mine them for features establishing the correlations needed for representing the corpus as a network. In Lori Emerson's terms (Emerson (2014)), the machine thus performs a reading-writing that maps every poem for the relevant features and also charts the whole corpus as informed by the interrelated quantifications of those features for all items (poems/texts) contained.

A poem accordingly evolves towards its own individuation by being integrated into the milieu which it shapes in its own turn by means of its own process, or performance, of reading-writing/writing-reading. As seen above, according to Simondon, this evolution can only take place by tapping into the 'raw' pre-individual 'matter' within and without the respective entity. In our case, as the poem is inscribed in digital media and digital space by and for computational processing algorithms, that said 'matter' consists of quite a number of layers, stages, parameters, and operations. Choices or default settings can be highly consequential for instance in terms of the kind of character encoding used (that makes 'common' alphabetical and possibly other types of characters readable to, and writable by, the machine) and other mechanisms of embedding a sequence of characters as text in digital media. Other settings or operations making for instance the file (and its path) accessible to the subsequent computational processing, or related to the algorithm(s) opening and reading the file (what is it they filter out or not in terms of characters, spaces, line-breaks, etc.) will impact significantly the processing and its output. At least equally consequential will be the choices related to the NLP sets and settings, the feature extraction and automated analysis (what kind of tokenization, what does the stop word set consist of, what kind of vectorization the script calls with what values for the parameters involved, etc). All of the above constitute poem's pre-individuation 'matter' or materials in digital media and digital space.[4]

It is such media and computation-related materials and factors that are instrumental

[4] In this particular case, the juxtaposition of the latter two environments—media and space (of the digital)—refers to a number of aspects among which the fact that since in digital culture, or the "culture of connectivity" (Van Dijck (2013)), nothing and nobody ever actually goes 'off the grid' (cf. for instance (Kirschenbaum, 2016)), files written and/or processed on personal/self-contained machines/directories are always infiltrated (when not totally generated and contained) by elements (apps, word processors, programs, coding libraries, etc.) based in digital space, and therefore actually totally immersed in the latter given its operability and ontology. Moreover, both the 'commons' (or 'base of operations')—JupyterHub—and the 'performance venue'—Twitter—used for that particular event are literally Web-based.

in inscribing and performing the poem in digital space and in its contribution to the generation and (re)shaping of its milieu and/as itself. The resulting milieu of poems and texts represent just a few levels of that multilayer Umwelt in which all of those computational and medial elements are active (and re/de-)formed in their turn. An important aspect here, and as already mentioned above, sensibly different from Stefans's vision (via Simondon), is the role of computational features in processing the poems and/into their poetry corpus milieu. In equating the poem with a technological individual, Stefans also translates the particulars of the latter's interaction with the environment, as reflected on its constitutive elements, onto the poem as well. As already briefly explained, this positions the said elements very much like the "heart or liver in the body" (Stefans (2017), p. 63) and therefore, and perhaps most importantly, as having "no associated milieu" and "not interacting with an environment outside of the machine" (Stefans (2017), idem). The machine here is, of course, the technological individual or, in Stefans's extrapolation, the poem. Still, in computing data features for the poem's and the corpus' processing and analysis it is exactly the constitutive (technological) elements of the poem that enact its interaction with, and shaping of (while being shaped by), the milieu. If in one layer of the multiplex network, for instance, we represent correlations between the tf-idf vectors of each and every poem, that will reflect the frequency of certain terms in each of the poems as measured against their frequency across the corpus. Then, if in another one, we represent the correlations between the vectors representing rhyme and euphony scores for every poem, that will reflect the intimate sonic anatomy of all poems and the types and density of rhymes and other euphonious devices across the corpus. It is not only that the heart and the liver of the poem interact with the environment, it is actually through them alone (and its other 'internal organs') that the poem interacts with, inhabits, shapes, and is shaped by, the environment.

There is, on the other hand, significant hybridization between the poetic and the non-poetic in the traditional sense of the word—or perhaps, apoetic—in the above outlined processes. In generating the graph, a tf-idf layer may be significant but definitely applies to any other genres and types of text as well. Also, as new files are added by participants to the dataset, all sorts of text enrich the corpus, and the poetic-feature classifiers run on them will just provide irregular or erratic output. Other poetry-driven concerns will impact the NLP framework and results.

For instance, we significantly tempered with the Python NLTK (natural language toolkit)[5] stop word set as we wanted the first personal pronoun not to be flushed out in tokenizing the texts. The first person singular is traditionally—although at times rather stereotypically—seen as regularly frequent in lyric poetry. Yet while certain poets can indeed use it really frequently, others can consistently, sometimes even blatantly, avoid it. Among the former there will be an iconic lyric poet such as Sappho, to pick just a most famous example. But keeping the 'I' in will definitely help to identify the most representative of the latter category as well. Emily Dickinson or Georg Trakl, for instance, if in the corpus, will strikingly stand out. Dickinson's metaphysical compression and symbolic scope goes beyond the individual in gnostic and gnomic ways, while Trakl's expressionist visions are either too alienating or too encompassing and ecstatic to allow or care about any 'I' submerged in them. We also chose not to consider the first person plural a stop word. Its occurrence, particularly as weighed against the singular, can suggest a nuanced and more or less self-equated take on the community, as is the case with the poetries of Walt Whitman or Lyn Hejinian.

[5] www.nltk.org

We did not want adverbs of place either to be counted for stop words especially since having a special interest in place poetry. But while the concern above can have a more general genre-related relevance and impact, this latter one was more of a task specific one. Besides being the world-renowned institution, DHSI also means to the members of the relevant communities, quite a number of personal and locative aspects that very interestingly complement its explicit and perhaps prevailing digital media and digital space focus. There are its charismatic and intellectually outstanding organizers and there is also the indelibly picturesque location, Victoria, B.C. Our option regarding such stop words was therefore meant not only to ensure the importance of place as event venue and genre-relevant stylistic feature at the same time, but to also capture intra- and inter-textual characteristics or ambiguities related to being digital space based and specifically geographically located at the same time. Most importantly, perhaps, to unveil textual corpus-imbedded features involved in distinguishing between the two and/or perceiving them as inextricably interrelated.

From a more general perspective, such fine-tuning of the stop word set (just as a host of other NLP-relevant choices and specific approaches) profoundly inform the inscription, processing, and analysis of poems within corpora. While this reflects a 'philosophy' of poetry reading, perhaps a poetics, on the part of the programmer, it does so in ways that most intimately fuse the computational and the poetic: the poems are the mode in which the machine reads and writes them, while the machine is *the* milieu of the poem. Such corollary stemming from NLP, and more generally, computational programming instrumentality will help us circle back to our main tenet regarding 'the beyond' of poetry as the poem's Umwelt and the humans-poems-machines ontological and operational commonality it implicates.

There are in that respect two main aspects to note regarding the inscriptive performance of poems (and texts networked into 'graph poems') in digital space and media. First, it is, in Simondon's (and post-Uexkul) philosophical, as well as Stefans's literary theoretical, terms here revisited from our own angle, precisely the technical elements that prevent the poem from becoming or remaining a technical individual and that *are* yet instrumental in its individuation. The technical—medial and computational—'apoetic' features and procedures that shape it as a poem in its digital milieu. 'Apoetic' here has a twofold tenure, technological data-science features and approaches that push or ignore the established/'traditional' boundaries of the poetic genre, and also, technology that involves no *poiesis*, no putting forth of any coherent self-contained 'individual' or 'oeuvre'. And, second in the above initiated enumeration, the technics making up that specific digital milieu of the poem undergo in their turn a process of individuation, as poetic commoning technology.

The sociality of poems and technologies is already there with its hybridizing and performative nature even before explicitly involving them in a 'performance' event. And they are there with their live (as in both living and at the actual time of occurrence/performance)—if not human—component already as well, as, according to Stefans via Simondon, "continuation of life 'by means other than life' " (Stefans (2017), p. 69). Even before engaging in a poetry-technology-driven commoning event, the poem is already individuated/ing as machinic and human when impacting humans and the interconnectivity between humans, just as the latter have already been re/de-formed by, and into, technologies or poems. That is where we find the social impact of poetry and/as technology at one of its possible peaks, with individuation and milieu involving the processually overlapping humans (as technological and poetic beings), poems (as performative inscriptions in digital space), and machines (as computationally implemented algorithms and artificial intelligence).

A previous initiative whereby we sought to demonstrate and enact the societal relevance and impact of computational poetry (and which the DHSI event actually built on) was the computationally assembled poetry anthology *"US" Poets Foreign Poets* (MARGENTO (2018)). While starting off by representing and analyzing an initial editorially selected corpus of contemporary U.S. poetry as a network graph the anthology advanced by algorithmically expanding it with poems that met certain diction-related criteria and, consequently, held certain peculiar positions in the gradually enlarged graph (cf. MARGENTO (2018)). Those newly added poems were included strictly based on the above mentioned features —having therefore a diction that conferred them certain topological prominence in the network—and irrespective of the author's region, thus opening the selection to anybody whose poem(s) fit the unusual profile, and translating the "U.S" in the title into "us," poets elsewhere and anywhere. This societal explicit implication and subversion of customary editorial politics of exclusion was noticed by critics and practitioners both on the social-literary level—Christopher Funkhouser, for instance, posed the rhetorical question "who among us ever dreamed that we'd see an anthology where Alan Sondheim's work resides near that of Charles Wright and Rita Dove, and in fact gets more page space than they do?" (Funkhouser (2019))—as well as the communal-ontological one (made evident by the "translation as process" and generative graph informing the collection), as John Cayley noted that the approach manages to transform "U.S. into 'US' because reading and translating in this way is what makes us us" (Cayley (2019)).

The DHSI 2019 event added to all of the above (and the book's challenging of medial essentialism inhabiting the current hardcoded gaps between print and digital) a 'commoning' approach involving live the community per se and resulting in a *webformance.* The corpus expanded by means of the corporeal and networked algorithmics straddled an online commons and a social media website it automatically inundated in ways that pushed the boundaries of corporate managed pre-established frameworks for 'user content'. In 'traditional' performance poetry the body is used as a medium of communication and/or status asserting prop, while physical artefacts are extensions of the body modulating the message and regulating (controlling even) the audience's reactions and/or participation in the performance (see for instance Novak (2011), pp. 151-169). The artefact in our case is the set of algorithms selecting the outstanding poems/texts contributed by the participants to the corpus (by means of network visualization and analysis), and sampling them (with further visualization and/or video-audio material) on Twitter. In doing so, the artefact contributes to generating a societal milieu, by bringing about "quasi-Umwelts," while to "a conscious observer" such "object comes to appear as if it generates an Umwelt" (Schwarz (2018), p. 66). Our conscious observers—or at least a part of them—were the very participants in the event and the milieu creation. The transition from the "quasi-Umwelt" above to what "appears" to be (i.e., is performed as) a full-fledged Umwelt marks in our view the participants' own transitioning from "observers" to being themselves performatively individuated. Their individuation was enacted by their involvement in the corpus expansion, the visualization and analysis of the resulting evolving network as collective assembly, and at times by being highlighted as remarkable co-assembler of the networked corpus whose contribution got sampled on Twitter.

We are not as naively utopian as to say that an experiment employing the artefacts above turn the traditional performance poetry paradigm on its head and thus, instead of controlling the audience, it liberates and includes them as full and unrestrained co-authors of the performance. Yet a couple of potential upsides could definitely be advanced here. While, for instance, in certain approaches to traditional performance it was more often than not strictly the performer's body that the corporeal dimension of the event resided in, and the artefacts were their exclusive props and emblems of authority/authorship, in our particular case, there is implicit but effective collective corporeal engagement with the corpus and the algorithms that thus become everybody's performative artefacts.

That kind of engagement begs a quick note on embodiment. According to N. Katherine Hayles, embodiment is (unlike the body) an always enmeshed and *contextual enactment* (cf. Hayles (2008), p. 196), and in our case, these elements are obviously demonstrated in the literal networked contextuality of the corpus and the algorithm, as well as in the performative enactment of individuation and milieu as mutually generative. Moreover though, within such an event the enactment is framed as *webformance,* that is, a performance that, in (re/de)forming the Web, has actually everything to do with processual corpus commoning. And it is therefore the process, the commoning, that gets embodied, amounting to a corpus corporeal embodiment that ranges from literal body evocations (as in sampling a graphic epigram from the *Shanzhai Lyric* corpus) or displays (as in pulling Margento videos off of YouTube) to visualizing the network at a certain stage of collective engagement.

This latter aspect alludes to another possible advantage of such approaches. Namely, the attempted transition from the control-the-audience to the systemic-control-subverting-participatory-audience paradigm. The former refers to performances in which the 'onstage' performer(s) (tend or need to) control the audience in conventionally medium-oblivious approaches and settings; the latter to events highlighting the inherent performative and at the same controlling nature of the medium and involving (sections of) the audience in commoning practices potentially subverting systemic control.[6] The cultural and political context for the latter is the typically the one of cultures of connectivity and digital space, where the anonymous system is inescapable and control is ubiquitous since embedded in the very medium and the networks per se (cf. for instance Franklin (2012), Tanasescu and Tanasescu (2021)). Therefore, while we could not realistically speak of escaping control, we can talk about exposing it and even working with control against control, a point we have developed in another forthcoming publication (Tanasescu (2021)). The strategy in our case involved going back and forth between two online platforms and using algorithms to automatically inundate one of them with the collective data assembled and analyzed on the other. Yet we are not simply talking about sampling in social media certain data stored elsewhere (although storing and generating data at a different web-based location accessible only to the participants was of crucial importance to the subversiveness of the experiment), nor about just creating a bot that will tweet samples off a given text/corpus (although the deployment of algorithms that output content and then tweet that content is also essential for the message and the performance experience). We are talking about *data commoning,* algorithmic community building through shared (collection of, work on,

[6] We are definitely not saying that the boundary between the two paradigms is the traditional-(post)digital one. It is actually imperiously necessary to note even if only in passing that 'page-based' poets have framed the potential social impact of poetry in terms strikingly similar to those in our discussion. Mainstream contemporary American poet and Kenyon Review editor David Baker recently stated, for instance, that "I do think poetry, the best of it, can help to shape a person's mind, his or her being-in-the-world, his receptiveness or her openness and rigor" (Baker and Quesada (2019)). Another awarded poet, Claudia Rankine talks about poetry's social function: "It's not arguing a point. It's creating an environment." (Baker and Quesada (2019))

and enlargement of) data, in/as *webformance,* performance that deploys networks (or user managed webs) to expose the Web's inherent control, which it de-forms and attempts to re-form by bypassing established fully regulated frameworks for online assembly and collective activities.

In conclusion, as poetry—and specifically the computational poetry work done within the *Graph Poem* project—has proved useful before in cross-disciplinary approaches that provided NLP and DH outcomes relevant beyond the genre-related tasks and even beyond the attendant literary concerns, the deployment and further development of those results in performance helped to cast light on a new dimension of such research, the social one. While the social relevance and potential impact of poetry is far from being the invention of our (post)digital society and our culture(s) of connectivity, new challenges, opportunities, and accordingly fine-tuned approaches emerged as possible in digital space and media. As outlined above, they mainly have to do with collective unconventional (alternative platform) data curation that will shape community in/as performance—*data-commoning*—and deploying network applications that disclose and subvert the ubiquitous control informing the Web by inscriptively and processually disrupting and deforming established frameworks for online social activity and assembly or, in one word, by means of *webformance*.

We drew in considering both the social-political and theoretical implications of such computational performance poetics on recent reframings of the concept of Umwelt as well as Simondon's philosophy of technics, particularly as revisited and adapted to poetry by Brian Kim Stefans. The concept of individuation—as inextricably intertwined with the one of milieu—turned out remarkably helpful in this endeavor. It is just that unlike Stefans for instance, our NLP and graph-theory-based approach to poems as inscriptive performances in digital space and media, while validating the fundamentally technological nature of poems, reached different conclusions regarding their interaction with, and generation of, their milieus. The technical elements making up a poem emerged therefore not only as not confined in the poem and away from the environment, but actually the very elements that enact its (non-individual) individuation and the shaping of its environment in both poetic and apoetic ways. The latter ensure in digital space and media the potentially pervasive social and community relevant impact of poetry in/as computational performance as it radically exposes and employs multilayered overlappings and interfusions between humans (as technological and poetic beings), poems (as performative inscriptions in digital space), and machines (as computationally implemented algorithms and artificial intelligence).

Funding

This work was supported by the Social Sciences and Humanities Research Council of Canada grant number 430-2014-00349.

References

Baker, D. and R. Quesada
2019. 4 contemporary poets on curiosity, generosity, and vulnerability.

Cayley, J.
2019. If i am a person, i make things with language. if i am a poet, i make art with language...

Emerson, L.
2014. *Reading writing interfaces: From the digital to the bookbound*, volume 44. U of Minnesota Press.

Franklin, S.
2012. Cloud control, or the network as medium. *Cultural Politics*, 8(3):443–464.

Funkhouser, C.
2019. My observations on *“us” poets foreign poets*.

Hayles, N. K.
2008. *How we became posthuman: Virtual bodies in cybernetics, literature, and informatics*. University of Chicago Press.

Kennedy, S.
2015. *Chaos media: a sonic economy of digital space*. Bloomsbury Publishing USA.

Kesarwani, V., D. Inkpen, S. Szpakowicz, and C. Tanasescu
2017. Metaphor detection in a poetry corpus. In *Proceedings of the Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, Pp. 1–9.

Kirschenbaum, M. G.
2016. *Track changes: A literary history of word processing*. Harvard University Press.

MARGENTO
2018. *“US” Poets Foreign Poets. A Computationally Assembled Anthology / Noi poeți „americani” poeți străini. Antologie asamblată computațional*. Fractalia.

Novak, J.
2011. *Live poetry: An integrated approach to poetry in performance*, volume 153. Rodopi.

Schwarz, J. A.
2018. Umwelt and individuation: Digital signals and technical being. In *Digital Existence*, Pp. 61–80. Routledge.

Stefans, B. K.
2017. *word toys: poetry and technics*. University of Alabama Press.

Tanasescu, C.
2021. The poetry and poetics of digital humanities. In *Digital Humanities in Canada*. University of Ottawa Press.

Tanasescu, C., V. Kesarwani, and D. Inkpen
2018. Metaphor detection by deep learning and the place of poetic metaphor in digital humanities. In *The Thirty-First International Flairs Conference*.

Tanasescu, C. and R. Tanasescu
2021. Text, code, control, and 'monstrous' iconicity. multilayer networks in digital media writing and the graph poem project. *Digital Studies / Le champ numérique*.

Tanasescu (MARGENTO), C.
2016. Community as Commoning, (Dis)Placing, and (Trans)Versing. From Participatory and 'Strike Art' to the Postdigital. *Dacoromania litteraria*, 3:10–44.

Van Dijck, J.
2013. *The culture of connectivity: A critical history of social media*. Oxford University Press.

Decomplexifying the network pipeline: a tool for RDF/Wikidata to network analysis

Julie M. Birkholz[1] and Albert Meroño-Peñuela[2]

[1]WeChangEd, Department of Literary Studies, Ghent University, Ghent, Belgium
[2]Department of Computer Science, Vrije Universiteit Amsterdam, The Netherlands

Knowledge Graphs that use the Resource Description Framework language (RDF) as a knowledge representation paradigm are increasingly popular in Digital Humanities, and represent a valuable source of data for network analysis. However, digital scholars interested in network approaches over RDF graphs have to deal with complex workflows and frameworks in order to perform their analyses. These complexities exacerbate complications in reproducing and replicating their work. In this paper, we detail a proof of concept to combine popular libraries in RDF data management and network analysis in one single, publicly accessible Jupyter Notebook that enables a structured approach to network analyses of RDF graphs. What sets our work apart specifically is its flexibility in quickly re-running network analyses over slightly modified RDF graphs, and ensuring transparency in making the code visible. We explain this approach through two case studies: women editors in Europe in the 19th century, and provenance of the harmonization of the historical Dutch censuses (1795-1971). This approach affords the researcher to quickly, easily, efficiently and with increased reliability project and analyse networks from RDF.

1 Introduction

Linked Data is an increasingly common way to publish structured data in the Humanities (de Boer et al., 2014, Meroño-Peñuela et al., 2015, Thornton et al., 2017). As Tim Berners-Lee, the "creator" of the Semantic Web, described - Linked Data “provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries.” ((W3C), 2011). Thus facilitating accessibility of knowledge on historical and cultural objects in a format readable by both humans and machines. For example, through standards such as the Resource Description Framework (RDF), natural language statements such as ‘‘`George Orwell wrote`

1984'' can be expressed as a triple consisting of: a subject (`:George_Orwell`), a predicate (`:wrote`), and an object (`:1984`). This knowledge can be retrieved by machines through a unique and global identifier (Uniform Resource Identifiers - URIs). This affords a networked archive, bringing together publicly available materials distributed in libraries, archives and museums; and thus allowing the researcher to integrate, and implement an unprecedented amount of often unstructured, siloed data, in lightning speed. Such an ontology or data model affords access, merging of information, and enrichment through efficiently linking of information on objects, entities and relations of collections to other collections.

Technically speaking data represented in the RDF language is structurally a graph. Thus it inherently allows us to infer relations, bundling any common affiliation between objects and attributes. From a research point of view this has led to a tendency to study RDF as a network. The study of networks and specifically the study of social networks has its roots in sociological theories where relationships form a part of the basis for understanding behavior(Durkheim, 1951, Simmel, 1955) where all actions are embedded in networks.(Granovetter, 1985) These relations – a set of edges, between nodes (entities) – define a network. These social networks reflect types of relations (e.g., a friendship tie in a friendship network or advice tie in an advice network). The study of networks, and in particular social networks, have been and are on the rise, providing explanations for relational and systematic phenomena(Borgatti and Foster, 2003), as it moves beyond explanations based on individual factors. For example not that someone's age explains their success, but rather the structure of their social network.(Granovetter, 1985)

The identification of networks is often thought of as a laborious task. It is traditionally done in many fields by searching through archival sources to identify nodes and edges, and reshaping data that is often not collected as relational, but from which one can infer relations. This entails integrating, and implementing a large amount of often unstructured, siloed and incomplete data to reconstruct relations between nodes and edges. Thus information about relations where a social network can be inferred from RDF provides a great advantage for exploring social networks embedded in this data. Networks can efficiently be reconstructed with the development of specific SPARQL queries to reflect different lenses of relations. For example, generating networks of different time periods, of different types of relations, with different boundaries (looking at relations of one city versus one country, or a neighborhood to a street) over the same data source.

Modeling data as networks affords the implementation of network analysis. Network analysis –the method used to analyze relations– provides a lens to investigate these diverse complex relational dynamics to examine structure, content or function. For social networks, which are the focus of the examples we provide in this paper, the structure of networks and positions of actors in these structures are seen as proxies for understanding social structure (Burt, 1980, Coleman, 1988).

The analysis of networks from RDF is largely done with a pipeline of tools (i.e. (Gil and Groth, 2011, Groth and Gil, 2011)). This starts with a data source, and the tools necessary for querying the specific data and relations. For example one workflow may be: the Wikidata Query Service, which allows one to query linked data in the Wikisphere through a SPARQL query, and can be exported in a number of formats; or the data might be stored in a database and is extract-able as a JSON(-LD) file. Moving from these file types, this relational data needs to converted into a file type that is readable by a network analysis software. Typical network analysis software use a range of inputs depending on the program. The two most commonly used user friendly network analysis and visualization programs with a graphic user interface are Gephi[1] and UCINet[2]. These programs allow the implementation of various types of input files; for example: .csvs, matrices and DL files, as well as program specific files. Then pending the required analysis there are a number of export options to further reuse these results as data. This, for example, could include analysing network measures and considering them as a variable in a statistical model in a program such as SPSS, or R. Thus the current workflow approaches for working with network data from RDF requires researchers to work through multiple programs to specify queries, extract networks and export data as matrices, and implement network analysis tools to investigate graphs.

In addition, in building such a pipeline we lose sight of the hermeneutics of the research objects.(Gibbs and Owens, 2013) Researchers are often faced with black boxed tools that limit their understanding of the projection, generation, analysis or reformatting that occurs with each step. With each use of an additional program, algorithm or command, the data gets re-"massaged" and shaped. This further becomes an issue, when the development of such a pipeline is a technical adversary for domain experts (e.g. historians, literary scholars) with (traditionally) limited technical knowledge; but also for researchers with specific expertise in RDF or networks. Thus, we argue there is a need, within the DH community, to reduce this RDF-to-network analysis pipeline without creating another domain or research question specific tool, and while maintaining oversight over the process from RDF-to graph-to network analysis.

To address these issues, we propose the use of a Jupyter notebook that integrates the Python packages: RDFLib[3] with NetworkX[4].[5] This results in a reusable workflow that allows network analyses over RDF data to be more accessible, flexible, transparent and iterative. This is due to that increases the reliability in exploring all the possible social networks within the available RDF, as well as increases the speed, ease, and efficiency of the necessary steps of RDF to network analysis. What specifically sets our work apart from previous workflows is its flexibility in quickly re-running network analyses over slightly modified RDF graphs, while maintaining the code visible for transparency and learning. We outline this pipeline through two case studies:

1. a case of the social networks of 19th century women editors in Europe available on Wikidata, and
2. provenance of the harmonization of the historical Dutch censuses (1795-1971)

to explain how it can be useful for humanities research.

2 Method

The notebook consists of five "cells", which are actionable code blocks, shown here in Figure 1. The output of all these processes can be selected and copy-pasted for further

[1] https://gephi.org/
[2] https://sites.google.com/site/ucinetsoftware/home
[3] https://github.com/RDFLib/RDFLib
[4] https://NetworkX.github.io/
[5] The full notebook is available at https://github.com/descepolo/rdf-network-analysis/blob/master/rdf-network-analysis.ipynb. A Google Colaboratory version of the notebook is also available, which makes it executable on the web with no need of local installation: https://colab.research.google.com/github/descepolo/rdf-network-analysis/blob/master/rdf-network-analysis.ipynb.

reuse in graph processing frameworks or directly in reports or papers.

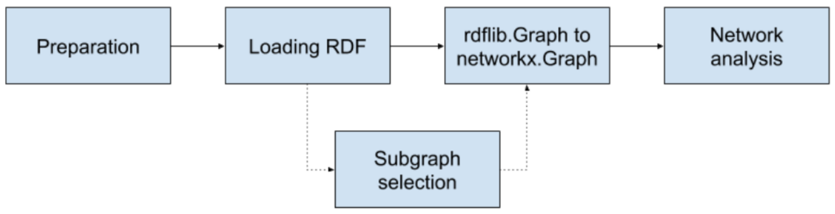


Figure 1: Workflow of the RDF Network Analysis Jupyter notebook

2.1 Preparation

As a first step the notebook loads the relevant packages - RDFLib and NetworkX. RDFLib is a Python package for working with RDF that includes parsers and serializers for RDF/XML, N3, NTriples, N-Quads, Turtle, TriX, RDFa and Microdata; a graph interface; store implementations for in memory storage and persistent storage on top of the Berkeley DB; and a SPARQL 1.1 implementation (Krech, 2006). This facilitates a flexible environment for loading and manipulating RDF graphs. Then the user is prompted to input the full path to an RDF graph to load the RDF graphs. This can be any local or online RDF file.

2.2 Subgraph Selection

Users select a specific network in the RDF graph. The efficient aggregation of different snapshots of the networks can be achieved through a SPARQL query. SPARQL is a Semantic Web query language for databases which enable the ability to retrieve and manipulate data RDF specifically (Segaran et al., 2009).

2.3 From RDFLib to NetworkX

In order to generate a network, this RDF needs to converted into a matrix. This is accomplished through a conversion of `RDFLib.Graph` to `NetworkX.Graph`. This prepares a file of the identified graph for analysis in NetworkX.

The Python library NetworkX enables the analysis of networks of around 10 million nodes and 100 million edges.(Hagberg and Conway, 2010) It is ideal for use for digital humanities as it affords the use of many types of networks, including directed graphs, and graphs with and self loops; while not maintaining strict object functions.(Hagberg et al., 2008) This implies that in the case of RDF which may have many and multiple types of networks embedded in the triples it will model anything that is structured as a matrices. This could include networks that we do not discuss here in this paper such as affiliation or two-mode networks, semantic networks and so forth. Thus the tool, which operates in the more general space of RDF models, does not limit the boundaries of inspection by imposing specific network models, leaving this choice to the user.

2.4 Network Analysis

Networks can be represented as graphs where positions and structures are systematically analyzed.(Wasserman et al., 1994) These principles originate from graph theory, which provides mathematical descriptions of characteristics.(Van Steen, 2010)

The networks can then be analyzed in NetworkX considering a number of characteristics of the network, as well as statistical analyses, see Table 1. Proposed Network Characteristics. We have selected a standard, non-exhaustive, set of one-mode complete network measures. This is to establish the proof of concept, of course in practice any network measure that is included in NetworkX could be implement in this notebook, for example measures of community detection, to other measures of centrality.[6] For a more exhaustive list and explanation of network measures see (Wasserman et al., 1994).

Following this selection the network analysis is run and the results are printed, as well as a basic visualization which serves for the researcher to confirm a first accuracy check of the network, e.g. were the correct node and edges selected?; does something look strange or potentially missed in the query?, that can now be amended.

Network Concepts	Network measures
network size	total number of nodes, and the average number of edges
power centrality	nodal position: e.g. degree centrality, betweenness, and eigenvector centrality (Freeman, 1978)
density	a value of the proportion of all possible ties that are present

Table 1: Network Characteristics.

3 Case Studies

In this Section we validate our approach using two different case studies for the Digital Humanities: the social networks of women editors in Europe in the 19th century; and the provenance graphs of harmonization transformations performed in the Dutch historical censuses. The use of these cases are to demonstrate the use of the notebook, not a network study with elaborated research questions and operationalized network measures.

3.1 Women Editors in Europe in the 19th century

The 19th century in Europe, was one of the onset and rise of industrialization, altered the socioeconomic and cultural norms influencing the movement of people through advancements in train infrastructure and technologies in food and consumer goods, and investments in education throughout Europe. This also led to an increasing advancement of women's rights and positions in society. The ERC "Agents of Change: Women Editors and Socio-Cultural Transformation in Europe, 1710-1920" (acronym WeChangEd) directed by Marianne Van Remoortel and based at the Department of Literary Studies, Ghent University, Belgium (project Agents of Change: Women Editors and Socio-Cultural Transformation in Europe, 2015), questioned how the press and periodical editorship in particular enabled women to take a prominent role in public

[6] It is not the goal of this paper to explain the operationalization of theoretical concepts to network measures, but it should be considered by humanities researchers in deciding on applicable measures to include in their research.

life, to influence public opinion and to shape transnational processes of change. To facilitate the collection of biographical records, and archival evidence of women editors in Europe a Linked Data model was developed (Schelstraete and Van Remoortel, 2019). This model afforded the cataloguing and tracing of different social networks in which the women participated.

This resulted in a large and growing database which includes 1700+ persons, 1600+ periodicals and 200+ organizations, as well as biographical information of these entities and relations between them, as identified through archival research. This data is available as the WCD Database, as subsets of data stored as .csv (Van Remoortel et al., 2020). In April 2020, the WCD database was imported to Wikidata (Thornton et al., ming) to facilitate the reuse and integration of this information with other Linked Open Data sources. The WeChangEd data can be identified in Wikidata through the unique property instance of WeChangEd ID P7947, see - `https://www.wikidata.org/wiki/Property:P7947`. This resulted in 3661 instances of data which compromises people, periodicals, organizations, as well as records of the relationships between these three entities, biographical information about these instances, and so forth. The complete dataset can be found via a Wikidata Query Service via - `https://w.wiki/QiQ`.

Identifying historical social networks is a laborious task, thus having the information on relations in Wikidata, and specifically as RDF, allows the researcher to explore historical social networks of the past in a more valid and flexible manner. The validity is increased, as the information is shared with the community, where it can be cross-checked, questioned, and enriched through the edit functions of Wikidata. As we show here through this example, the flexibility is affording by this pipeline.

In exploring how a researcher can identify social networks of these editors we display here three examples of projecting personal relationships of female editors between individuals as identified within the WeChangEd dataset. To identify these relationships we developed three SPARQL queries for the Wikidata Query Service, which we detail here below, and are also available at: `https://w.wiki/Qtr`, `https://w.wiki/QiQ`, and `https://w.wiki/QcS`, respectively. Using these graphs as input for the method described in Section 2, we convert these graphs to a NetworkX file and the network analysis is executed. We implement this query in the notebook resulting in three different network projections, and reflect on the implications for digital humanities researchers in compiling social networks from the past.

The first network represents a query on the entire WCD dataset, to identify kinship relations, this includes any identified siblings, parents, unmarried partner, spouse, or children of female editors `https://w.wiki/Qtr` (see Listing 1).

This results in a network of all female editors and their relationships as identified in Wikidata, where nodes are individuals and edges or ties of represent a personal relationship, see Figure 2.

```
SELECT DISTINCT ?item ?o ?itemLabel ?sibling ?spouse ?partner ?father
?mother ?child
WHERE

{
# find occupation editors
?item wdt:P106 wd:Q1607826.
?item wdt:P7947 ?o .

# that are female
?item wdt:P21 wd:Q6581072.

# that have a birth and death date
?item wdt:P569 ?birthDate.
?item wdt:P570 ?deathDate .

# with kinship: sibling
OPTIONAL { ?item wdt:P3373 ?sibling .}
# with kinship: spouse
OPTIONAL { ?item wdt:P26 ?spouse .}
# with kinship: unmarried partner
OPTIONAL { ?item wdt:P451 ?partner .}

# with kinship: father
OPTIONAL { ?item wdt:P22 ?father .}
# with kinship: mother
OPTIONAL { ?item wdt:P25 ?mother .}
# with kinship: child
OPTIONAL { ?item wdt:P40 ?child .}

# labels
SERVICE wikibase:label { bd:serviceParam wikibase:language
"[AUTO_LANGUAGE],en". }

} ORDER BY ?birthDate ?deathDate
```

Listing 1: SPARQL query for all female authors and their kinship relations

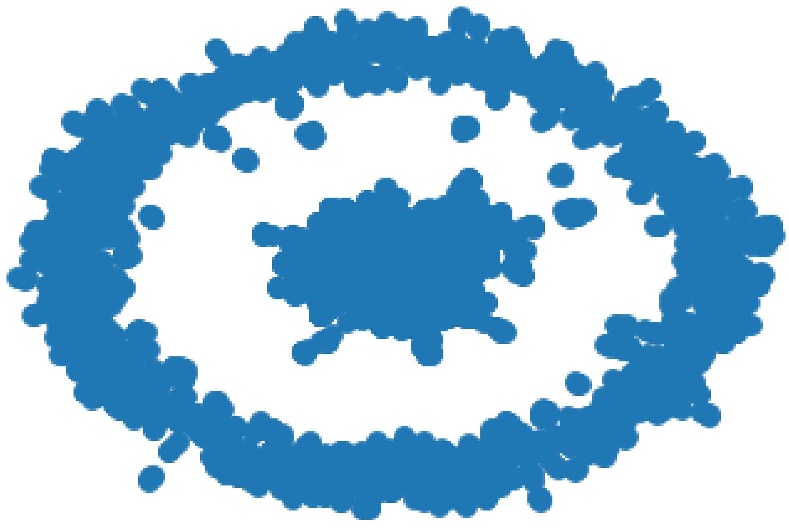


Figure 2: Network of editors

```
SELECT DISTINCT ?item ?o ?itemLabel ?sibling ?spouse ?partner ?father
?mother ?child
WHERE

{
# find occupation editors
?item wdt:P106 wd:Q1607826.
?item wdt:P7947 ?o .

# that are female
?item wdt:P21 wd:Q6581072.

# that have a birth and death date
?item wdt:P569 ?birthDate.
?item wdt:P570 ?deathDate.

 # that is British
?item wdt:P27 wd:Q174193.

# with kinship: sibling
OPTIONAL { ?item wdt:P3373 ?sibling .}
# with kinship: spouse
OPTIONAL { ?item wdt:P26 ?spouse .}
# with kinship: unmarried partner
OPTIONAL { ?item wdt:P451 ?partner .}

# with kinship: father
OPTIONAL { ?item wdt:P22 ?father .}
# with kinship: mother
OPTIONAL { ?item wdt:P25 ?mother .}
# with kinship: child
OPTIONAL { ?item wdt:P40 ?child .}

# only active in the 19th century
FILTER ( ?birthDate >= "1800-01-01T00:00:00Z"^^xsd:dateTime &&
?deathDate <= "1898-12-31T00:00:00Z"^^xsd:dateTime )

# labels
SERVICE wikibase:label { bd:serviceParam wikibase:language
"[AUTO_LANGUAGE],en". }

} ORDER BY ?birthDate ?deathDate
```

Listing 2: SPARQL query for relationships of British female editors of periodicals in the 19th century in Wikidata

In this second selection we aim to show, how to refine the query, to select a more bounded set of nodes. This is a bounded selection of relations from within the WCD dataset but specifically of 19th century British female editors and their kinship relations, this includes any identified siblings, parents, unmarried partner, spouse, or children: `https://w.wiki/QnA` (see Listing 2).

This results in a network of the personal relations of 19th century British female editors, where nodes are individuals and edges are relationships, see Figure 3. This network is a subset of the larger graph, but with parameters of time - editors living during the 19th century, and place -what was then the United Kingdom of Great Britain and Ireland.

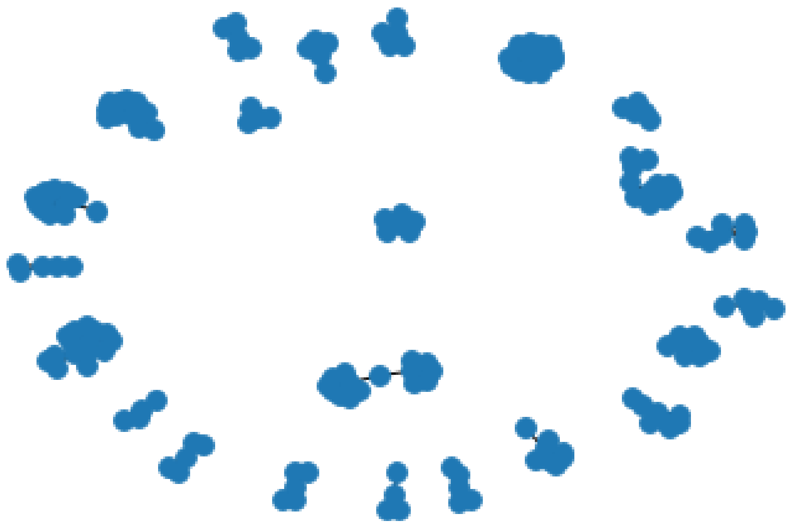


Figure 3: Network of 19th Century British female editors

The third case, aims to represent a different subset of the data, that is looking at relationships between editors based on language, instead of a geographical or political space. This selection represents a query of 19th century German speaking female editors and their kinship relations, this includes any identified siblings, parents, unmarried partner, spouse, or children:- `https://w.wiki/QnB` (see Listing 3).

This results in a network of relations of German speaking female editors as identified on Wikidata. Selecting German speaking instead of a specific empires and or nation-state provides a broader query for identifying possible interactions between the German-speaking community in the 19th century. This results in a network of individuals as nodes and edges as relations, see Figure 4.

The complete results for two specific social networks of 19th century British female editors and 19th century German speaking female editors can be found in detail in the appendix. The results show the network analysis on connected components or groups of connected individuals, most central nodes, and communities. A researcher can use these results, combined with the visualizations to further explore these relations either returning to archival materials to investigate previously understudied relations, or further analyse the structure and positions within these network to explain social capital of the periodicals the editors edited or kinship relations.

These three examples from within the WCD dataset on Wikidata display the flexibility of this approach in moving through a dataset, to generate social networks. This notebook, in contrast to other workflows allows researchers to consider aspects of space, time, place and other parameters of the data within a few steps and seconds; where the researcher can move and back and forth between the raw data, the query, the network projection, and the analysis, to compile the most suitable, reliable graph from the available data. It serves as both an efficient approach to explore the social relations within a dataset, as well as to validly and reliably generate a social network and conduct social network analysis of the networks from RDF.

```
SELECT DISTINCT ?item ?o ?itemLabel  ?sibling ?spouse ?partner ?father
?mother ?child
WHERE

{
# find occupation editors
?item wdt:P106 wd:Q1607826.
?item wdt:P7947 ?o .

# that are female
?item wdt:P21 wd:Q6581072.

# that have a birth and death date
?item wdt:P569 ?birthDate.
?item wdt:P570 ?deathDate.

# that speaks German
?item wdt:P1412 wd:Q188.

# with kinship: sibling
OPTIONAL { ?item wdt:P3373 ?sibling .}
# with kinship: spouse
OPTIONAL { ?item wdt:P26 ?spouse .}

# with kinship: father
OPTIONAL { ?item wdt:P22 ?father .}
# with kinship: mother
OPTIONAL { ?item wdt:P25 ?mother .}
# with kinship: child
OPTIONAL { ?item wdt:P40 ?child .}

# only active in the 19th century
FILTER ( ?birthDate >= "1800-01-01T00:00:00Z"^^xsd:dateTime &&
?deathDate <= "1898-12-31T00:00:00Z"^^xsd:dateTime )

# labels
SERVICE wikibase:label { bd:serviceParam wikibase:language
"[AUTO_LANGUAGE],en". }

} ORDER BY ?birthDate ?deathDate
```

Listing 3: SPARQL query for relationships of German speaking editors of periodicals in the 19th century in Wikidata

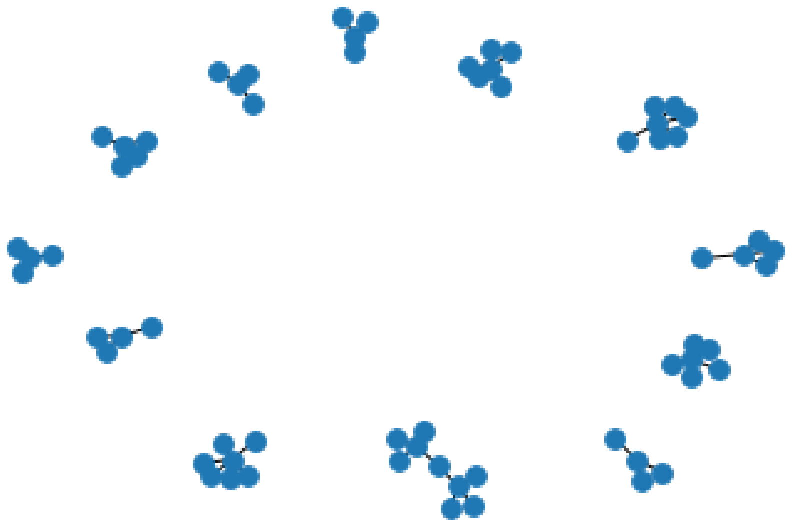


Figure 4: Network of 19th Century German speaking female editors

3.2 CEDAR: Harmonization Provenance of the Dutch Historical Censuses (1795-1971)

The Dutch historical censuses were collected in the Netherlands in the period 1795–1971, in 17 different editions, once every 10 years. The government counted all the country's population, door-to-door, and aggregated the results in three different census types: demographic (age, gender, marital status, location, belief), occupational (occupation, occupation segment, position within the occupation), and housing (ships, private houses, government buildings, occupied status). After 1971, this exhaustive collection stopped due to social opposition, and the government switched to municipal registers and sampling (Ashkpour et al., 2015). Various projects have digitized the resulting census data (CBS; IISH; Data Archiving and Networked Services[7], DANS; and the Netherlands Interdisciplinary Demographic Institute[8], NIDI), and have manually translated them into a collection of 507 Excel spreadsheets and 2,288 census tables[9]. The CEDAR project[10] takes these spreadsheets as input, and produces a Knowledge Graph of 6.8 million statistical observations (Meroño-Peñuela et al., 2015) many of which went through an harmonization process to satisfy the standardization needs of historians for their querying (Ashkpour et al., 2015).

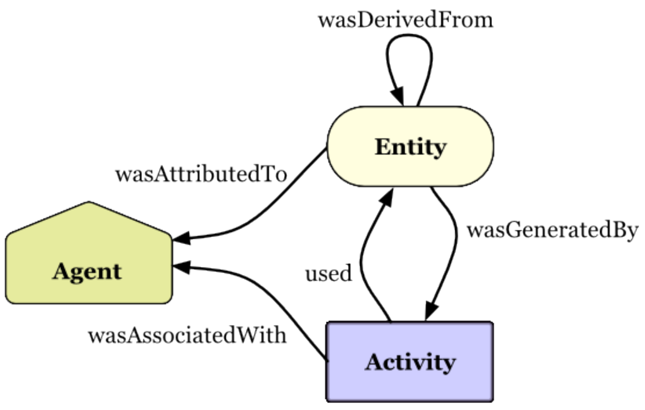


Figure 5: Provenance model of W3C PROV (Lebo et al., 2013).

In this case study, we use the CEDAR Knowledge Graph (Meroño-Peñuela et al., 2015) with our proposed approach to explain how to a researcher can consider network similarities and differences between various historical census data points and their *provenance* information. Historians are particularly interested in the transformation and manipulations that occurred in this harmonization process in generating these data points; as this signals their correctness and hence its reliability. Fortunately, the CEDAR Knowledge Graph documents the harmonization transformations of all data points using the W3C PROV standard (Lebo et al., 2013). This standard models provenance as the interactions between various *entities* (the objects subject to transformations, i.e. the census data points), *activities* (the transformation processes themselves, i.e. the harmonization rules) and *agents* (the persons or programs commanding the transformations) as shown in Figure 5.

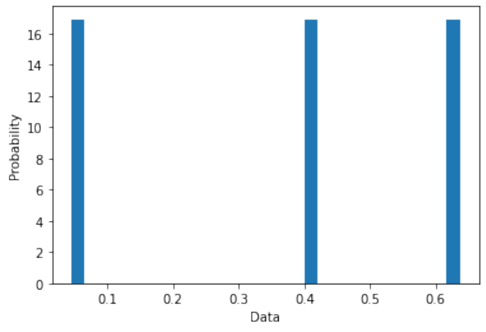
We select two arbitrary observations of the census, `VT_1859_01_H1-S8-J647-h` (observation 1, o_1) and `VT_1920_01_T-S0-R10108-h` (observation 2, o_2), and their corre-

[7] See http://www.dans.knaw.nl/
[8] http://www.nidi.knaw.nl/en/
[9] http://volkstellingen.nl/
[10] https://www.cedar-project.nl/

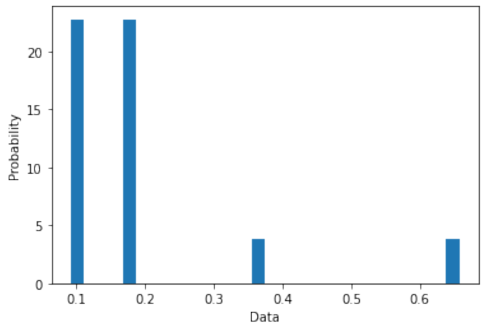
```
CONSTRUCT {
 ?obs ?obs_p ?obs_o .
 ?act ?act_p ?act_o .
} WHERE {
 VALUES ?obs {:VT_1859_01_H1-S8-J647-h :VT_1920_01_T-S0-R10108-h}
 ?obs prov:wasGeneratedBy ?act .
 ?obs ?obs_p ?obs_o .
 ?act ?act_p ?act_o .
}
```

Listing 4: SPARQL query for the harmonization provenance graphs of two census observations.

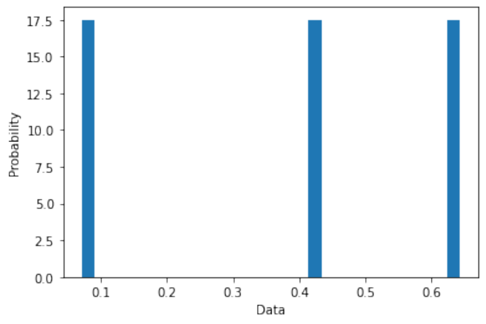
sponding provenance traces with the query shown in Listing 4 against the CEDAR SPARQL endpoint[11]. We use the graphs returned by this query as input for the notebook.[12]



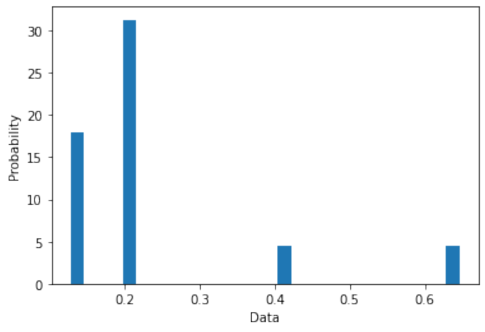
(a) o_1 degree centrality.



(b) o_1 eigenvector centrality.



(c) o_2 degree centrality.



(d) o_2 eigenvector centrality.

Figure 6: Histogram plots of o_1 and o_2 degree and eigenvector centrality.

We use the provenance graphs of o_1 and o_2 as input for the method described in Section 2. We execute the preparation block; we use the query of Listing 4 as subgraph selection; we execute the network conversion block; and finally we execute the network analysis block. The output networks as plotted by the notebook are shown in Figure 7. We can observe that for both cases the network is 2-star shaped, with the nodes representing the observation and the activity at the center of these stars and various nodes describing their properties, as expected. One edge (`prov:wasGeneratedBy`) connects these two nodes. An noticeable difference is that while o_1 (Figure 7a) is

[11] https://api.druid.datalegend.net/datasets/datalegend/CEDAR-S/services/CEDAR-S/sparql

[12] The input graphs are also available at https://github.com/albertmeronyo/rdf-network-analysis/blob/master/uc1.nt and https://github.com/albertmeronyo/rdf-network-analysis/blob/master/uc2.nt

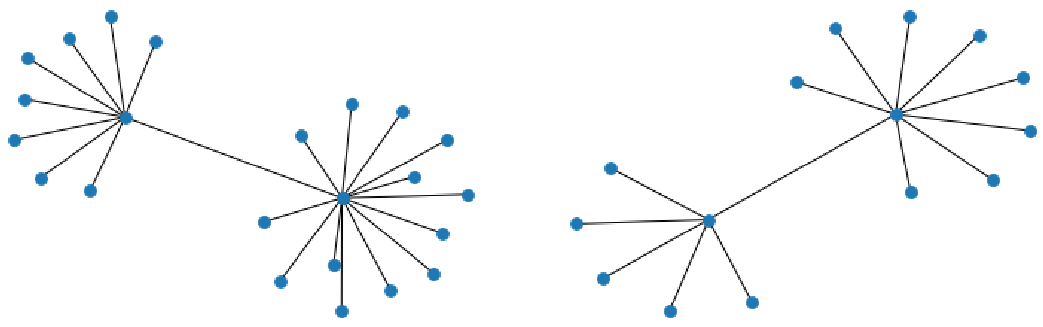

(a) Network of o_1 (b) Network of o_2

Figure 7: Network plots of two CEDAR observations and their harmonization provenance traces.

transformed by 6 different harmonization rules, o_2 (Figure 7b) is only affected by 3. This can provide interesting insights for historians, who may be keen to examine statistical observations that have been subject to a higher number of transformations (and therefore more prone to errors) and the relations of these transformations to their immediate context. In this sense, visualizing these network contexts can be a powerful tool for interpretation.

Additionally, Figure 6 shows the histograms for degree and eigenvector centrality drawn by the notebook for both graphs. This is a more aggregated view on the networks, showing similar behaviour for o_1 and o_2 (due to the structural similarity of provenance graphs) but also interesting differences. For example, o_1 eigenvector centrality shows a more normal distribution due to the higher variety of node influence in a more varied network. The remaining network statistics can be found upon the execution of these two examples in the notebook at `https://github.com/descepolo/rdf-network-analysis/blob/master/rdf-network-analysis.ipynb`.

4 Conclusion and Future Work

In this paper we have detailed how we have proposed to combine popular libraries in RDF data management and network analysis in one single, publicly accessible Jupyter Notebook that enables a structured approach to network analyses of RDF graphs. With the proposed Juypter notebook we have developed a transparent and iterative tool for RDF to network in research. The open code and user-friendliness of the notebook ensures flexibility for users in implementing different aspects of the two libraries that we did address here in this demonstration. In addition, we have demonstrated through the tool and use cases how this affords the reuse and accessibly for non-technical scholars of RDF, as well as increase the efficiency and flexibility of use for generating networks from RDF. This approach facilitates the study of diverse types of networks from RDF and thus study of relational phenomenon in the Humanities and beyond.

In addition, this approach proves, contrary to the trend in the digital humanities, that we do not need a new network software that converts diverse file types to make fundamental improvements on both the quality of the networks used in research, as well as the the analysis of networks. Rather, as we presented, a fundamental rethinking of how data on social networks is structured, manipulated and pushed through a pipeline is needed to efficiently generate, project and evaluate networks.

This approach increases the flexibility, compared to traditional network workflows- where the analyst would prepare a matrix for each projection of a network, go back to source material every time to reshape the data and networks based on different periods, or parameters (e.g. variables such as country of birth, gender, language of entities), and push it through the workflow. Such an approach reduces the technical adversary of knowledge on RDF and network analysis, while avoiding a black boxed software, as well as retains a hermeneutic approach to the source data, allowing the researcher to iteratively and efficiently requery, reshape and reanalyze the networks embedded in RDF.

Acknowledgements

The authors would like to thank Prof. dr. Paul Groth for his feedback on this work but also for his ongoing support of all of our crazy ideas. The first author would like to thank her partner- Radboud Reijn for being so patient, and secondly the continued support from Prof. dr. Marianne Van Remoortel in proposing and believing in a DH project for periodical studies. The work of the first author was supported by the H2020 European Research Council under the ERC Starting Grant agreement no. 639668. This work has been partly supported by the Dutch national project CLARIAH.

References

Ashkpour, A., A. Meroño-Peñuela, and K. Mandemakers
2015. The aggregate Dutch historical censuses: Harmonization and RDF. *Historical Methods: A Journal of Quantitative and Interdisciplinary History*, 48(4):230–245.

Borgatti, S. P. and P. C. Foster
2003. The network paradigm in organizational research: A review and typology. *Journal of management*, 29(6):991–1013.

Burt, R. S.
1980. Models of network structure. *Annual review of sociology*, 6(1):79–141.

Coleman, J. S.
1988. Social capital in the creation of human capital. *American journal of sociology*, 94:S95–S120.

de Boer, V., M. van Rossum, J. Leinenga, and R. Hoekstra
2014. Dutch ships and sailors linked data. In *International Semantic Web Conference*, Pp. 229–244. Springer.

Durkheim, E.
1951. Suicide: A study in sociology (ja spaulding & g. simpson, trans.). *Glencoe, IL: Free Press.(Original work published 1897)*.

Freeman, L. C.
1978. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239.

Gibbs, F. and T. Owens
2013. The hermeneutics of data and historical writing. *Writing history in the digital age*, 159.

Gil, Y. and P. Groth
2011. LinkedDataLens: linked data as a network of networks. In *Proceedings of the sixth international conference on Knowledge capture*, Pp. 191–192. ACM.

Granovetter, M.
1985. Economic action and social structure: The problem of embeddedness. *American journal of sociology*, 91(3):481–510.

Groth, P. and Y. Gil
2011. Linked data for network science. In *Proceedings of the First International Conference on Linked Science-Volume 783*, Pp. 1–12. CEUR-WS. org.

Hagberg, A. and D. Conway
2010. Hacking social networks using the python programming language. *Sunbelt 2010, Riva del Garda, Italy*.

Hagberg, A., P. Swart, and D. S Chult
2008. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

Krech, D.
2006. Rdflib: A python library for working with rdf.

Lebo, T., S. Sahoo, D. McGuinness, K. Belhajjame, J. Cheney, D. Corsar, D. Garijo, S. Soiland-Reyes, S. Zednik, and J. Zhao
2013. Prov-o: The prov ontology. *W3C recommendation*.

Meroño-Peñuela, A., A. Ashkpour, C. Guéret, and S. Schlobach
2015. CEDAR: the Dutch Historical Censuses as Linked Open Data. *Semantic Web*, 8(2):297–310.

project Agents of Change: Women Editors, E. and .-. a. W. Socio-Cultural Transformation in Europe
2015. Wechanged.

Schelstraete, J. and M. Van Remoortel
2019. Towards a sustainable and collaborative data model for periodical studies. *Media History*, 25(3):336–354.

Segaran, T., C. Evans, and J. Taylor
2009. *Programming the Semantic Web: Build Flexible Applications with Graph Data*. " O'Reilly Media, Inc.".

Simmel, G.
1955. The web of group-affiliations. conflict and the web of groupaffiliations.

Thornton, K., J. M. Birkholz, M. V. Remoortel, and S.-N. Kenneth
forthcoming. Working paper on bringing to light to women editors of the past through linked open data on wikidata.

Thornton, K., E. Cochrane, T. Ledoux, B. Caron, and C. Wilson
2017. Modeling the domain of digital preservation in wikidata. In *Proceedings of ACM International Conference on Digital Preservation, Kyoto, Japan*.

Van Remoortel, M., J. Birkholz, J. Schelstrate, M. Alesina, C. Bezari, C. D’Eer, E. Forestier, N. De Grave-Geeraert, M. Goethals, J. A., and G. Vanhulle
2020. Wechanged database.

Van Steen, M.
2010. Graph theory and complex networks. *An introduction*, 144.

(W3C), W. W. W. C.
2011. W3c semantic web activity.

Wasserman, S., K. Faust, et al.
1994. *Social network analysis: Methods and applications*, volume 8. Cambridge university press.

A. 19th century British female editors and their kinship relations as present in Wikidata

May 15, 2020

1 Network Analysis of RDF Graphs

In this notebook we provide basic facilities for performing network analyses of RDF graphs easily with Python rdflib and networkx

We do this in 4 steps: 1. Load an arbitrary RDF graph into rdflib 2. Get a subgraph of relevance (optional) 3. Convert the rdflib Graph into an networkx Graph, as shown here 4. Get an network analysis report by running networkx's algorithms on that data structure

[13]:

1.1 0. Preparation

```
# Install required packages in the current Jupyter kernel
# Uncomment the following lines if you need to install these libraries
# If you run into permission issues, try with the --user option
import sys
# !pip install -q rdflib networkx matplotlib scipy
!{sys.executable} -m pip install rdflib networkx matplotlib scipy --user

# Imports
from rdflib import Graph as RDFGraph
from rdflib.extras.external_graph_libs import rdflib_to_networkx_graph
import networkx as nx
from networkx import Graph as NXGraph
import matplotlib.pyplot as plt
import statistics
import collections
```

```
Requirement already satisfied: rdflib in /home/amp/.local/lib/python3.8/site-
packages (5.0.0)
Requirement already satisfied: networkx in /home/amp/.local/lib/python3.8/site-
packages (2.4)
Requirement already satisfied: matplotlib in
/home/amp/.local/lib/python3.8/site-packages (3.2.1)
Requirement already satisfied: scipy in /home/amp/.local/lib/python3.8/site-
packages (1.4.1)
Requirement already satisfied: pyparsing in /usr/lib/python3/dist-packages (from
rdflib) (2.4.6)
Requirement already satisfied: six in /usr/lib/python3/dist-packages (from rdflib)
(1.14.0)
```

```
Requirement already satisfied: isodate in /home/amp/.local/lib/python3.8/site-
packages (from rdflib) (0.6.0)
Requirement already satisfied: decorator>=4.3.0 in /usr/lib/python3/dist-
packages (from networkx) (4.4.2)
Requirement already satisfied: kiwisolver>=1.0.1 in
/home/amp/.local/lib/python3.8/site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: numpy>=1.11 in /usr/lib/python3/dist-packages
(from matplotlib) (1.17.4)
Requirement already satisfied: cycler>=0.10 in
/home/amp/.local/lib/python3.8/site-packages (from matplotlib) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/lib/python3/dist-
packages (from matplotlib) (2.7.3)
```

1.2 1. Loading RDF

The first thing to do is to load the RDF graph we want to perform the network analysis on. By executing the next cell, we'll be asked to fill in the path to an RDF graph. This can be any path, local or online, that we can look up.

Any of the Turtle (`ttl.`) files that we include with this notebook will do; for example, `bsbm-sample.ttl`. But any Web location that leads to an RDF file (for example, the GitHub copy of that same file at https://raw.githubusercontent.com/albertmeronyo/rdf-network-analysis/master/bsbm-sample.ttl; or any other RDF file on the Web like https://raw.githubusercontent.com/albertmeronyo/lodapi/master/ghostbusters.ttl) will work too.

[14]:
```
# RDF graph loading
path = input("Path or URI of the RDF graph to load: ")
rg = RDFGraph()
rg.parse(path, format='turtle')
print("rdflib Graph loaded successfully with {} triples".format(len(rg)))
```

```
Path or URI of the RDF graph to load: wechanged-british.ttl
rdflib Graph loaded successfully with 155 triples
```

1.3 2. Get a subgraph out of the loaded RDF graph (optional)

This cell can be skipped altogether without affecting the rest of the notebook; but it will be useful if instead of using the whole RDF grahp of the previous step, we just want to use a subgraph that's included in it.

By executing the next cell, we'll be asked two things:

- The URI of the ''entiy" type we are interested in (e.g. `http://dbpedia.org/ontology/Band`)
- The URI of the ''relation" connecting entities we are interested in (e.g. `http://dbpedia.org/ontology/influencedBy`)

Using these two, the notebook will replace the original graph with the subgraph that's constructed by those entity types and relations only.

```
[ ]: # Subgraph construction (optional)
     entity = input("Entity type to build nodes of the subgraph with: ")
     relation = input("Relation type to build edges of the subgraph with: ")

     # TODO: Use entity and relation as parameters of a CONSTRUCT query
     query = """
     PREFIX bsbm: <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/vocabulary/>
     CONSTRUCT {{ ?u a {} . ?u {} ?v }} WHERE {{ ?u a {} . ?u {} ?v }}""".
      ↪format(entity, relation, entity, relation)
     # print(query)
     subg = rg.query(query)

     rg = subg
```

1.4 3. Converting rdflib.Graph to networkx.Graph

Thanks to the great work done by the rdflib developers this step, which converts the basic graph data structure of rdflib into its equivalent in networkx, is straightforward. Just run the next cell to make our RDF dataset ready for network analysis!

```
[15]: # Conversion of rdflib.Graph to networkx.Graph
      G = rdflib_to_networkx_graph(rg)
      print("networkx Graph loaded successfully with length {}".format(len(G)))
```

```
networkx Graph loaded successfully with length 174
```

1.5 4. Network analysis

At this point we can run the network analysis on our RDF graph by using the networkx algorithms. Exeucting the next cell will output a full network analysis report, with the following parts:

- General network metrics (network size, pendants, density)
- Node centrality metrics (degree, eigenvector, betwenness). For these, averages, stdevs, maximum, minimum and distribution histograms are given
- Clustering metrics (connected components, clustering)
- Overall network plot

The report can be easily selected and copy-pasted for further use in other tools.

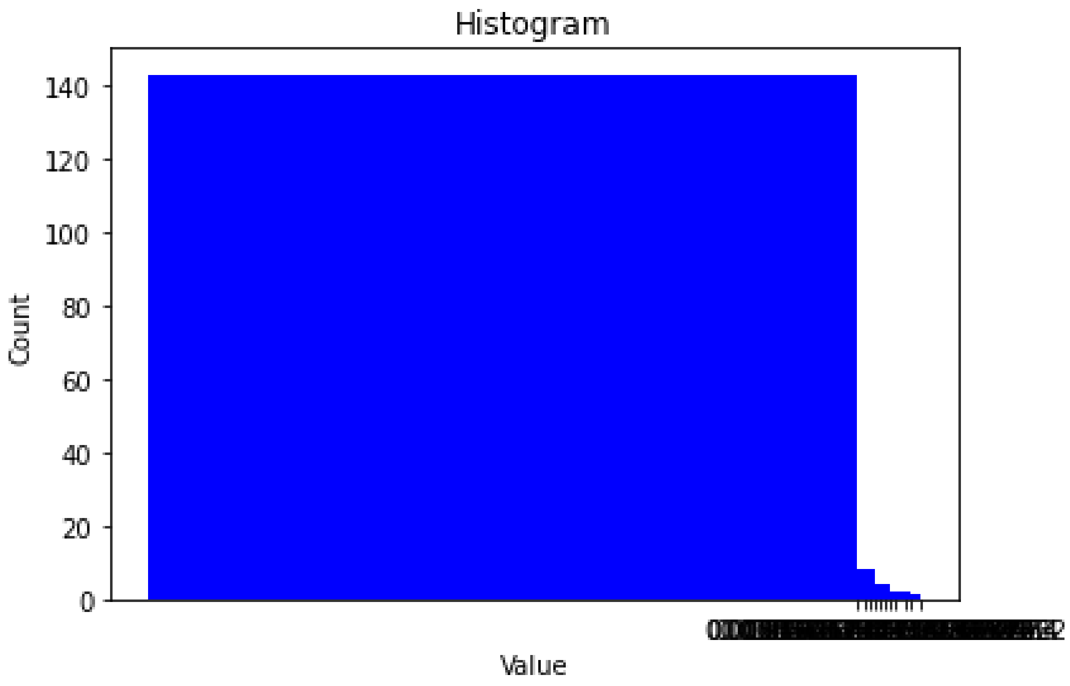
```
[17]: # Analysis

      def mean(numbers):
          return float(sum(numbers)) / max(len(numbers), 1)

      def number_of_pendants(g):
          """
          Equals the number of nodes with degree 1
          """
          pendants = 0
```

```
    for u in g:
        if g.degree[u] == 1:
            pendants += 1
    return pendants


def histogram(l):
    degree_sequence = sorted([d for n, d in list(l.items())], reverse=True)
    degreeCount = collections.Counter(degree_sequence)
    deg, cnt = zip(*degreeCount.items())
    print(deg, cnt)

    fig, ax = plt.subplots()
    plt.bar(deg, cnt, width=0.80, color='b')

    plt.title("Histogram")
    plt.ylabel("Count")
    plt.xlabel("Value")
    ax.set_xticks([d + 0.4 for d in deg])
    ax.set_xticklabels(deg)

    plt.show()

# Network size
print("NETWORK SIZE")
print("============")
print("The network has {} nodes and {} edges".format(G.number_of_nodes(), G.
 ↪number_of_edges()))
print()

# Network size
print("PENDANTS")
print("============")
print("The network has {} pendants".format(number_of_pendants(G)))
print()

# Density
print("DENSITY")
print("============")
print("The network density is {}".format(nx.density(G)))
print()

# Degree centrality -- mean and stdev
dc = nx.degree_centrality(G)
degrees = []
for k,v in dc.items():
    degrees.append(v)
```

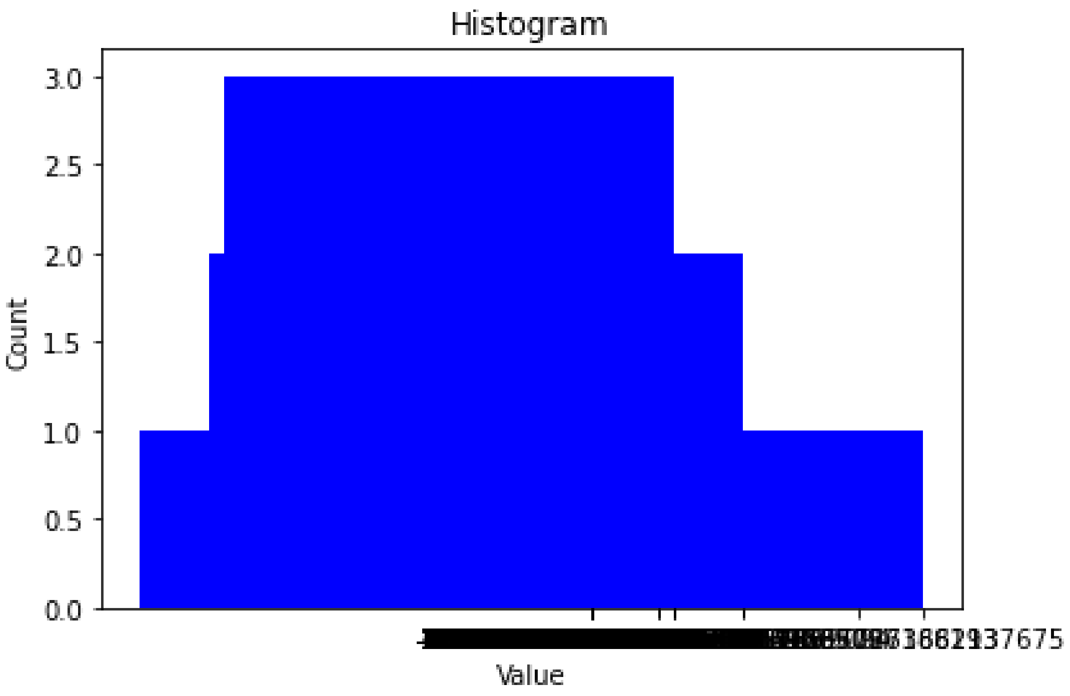
```
print("DEGREE CENTRALITY")
print("=================")
print("The mean degree centrality is {}, with stdev {}".format(mean(degrees),␣
↪statistics.stdev(degrees)))
print("The maximum node is {}, with value {}".format(max(dc, key=dc.get),␣
↪max(dc.values())))
print("The minimum node is {}, with value {}".format(min(dc, key=dc.get),␣
↪min(dc.values())))
histogram(dc)
print()

# Eigenvector centrality -- mean and stdev
ec = nx.eigenvector_centrality_numpy(G)
degrees = []
for k,v in ec.items():
    degrees.append(v)

print("EIGENVECTOR CENTRALITY")
print("======================")
print("The mean network eigenvector centrality is {}, with stdev {}".
↪format(mean(degrees), statistics.stdev(degrees)))
print("The maximum node is {}, with value {}".format(max(ec, key=ec.get),␣
↪max(ec.values())))
print("The minimum node is {}, with value {}".format(min(ec, key=ec.get),␣
↪min(ec.values())))
histogram(ec)
print()

# Betweenness centrality -- mean and stdev
# bc = nx.betweenness_centrality(G)
# degrees = []
# for k,v in bc.items():
#     degrees.append(v)
# print("BETWEENNESS CENTRALITY")
# print("======================")
# print("The mean betwenness centrality is {}, with stdev {}".
↪format(mean(degrees), statistics.stdev(degrees)))
# print("The maximum node is {}, with value {}".format(max(bc, key=bc.get),␣
↪max(bc.values())))
# print("The minimum node is {}, with value {}".format(min(bc, key=bc.get),␣
↪min(bc.values())))
# histogram(bc)
# print()
```

```python
# Connected components
cc = list(nx.connected_components(G))
print("CONNECTED COMPONENTS")
print("====================")
print("The graph has {} connected components".format(len(cc)))
for i,c in enumerate(cc):
    print("Connected component {} has {} nodes".format(i,len(c)))
print()

# Clusters
cl = nx.clustering(G)
print("CLUSTERS")
print("========")
print("The graph has {} clusters".format(len(cl)))
for i,c in enumerate(cl):
    print("Cluster {} has {} nodes".format(i,len(c)))
print()

# Plot
print("Visualizing the graph:")
plt.plot()
plt.figure(1)
nx.draw(G, with_labels=False, font_weight='normal', node_size=60, font_size=8)
plt.figure(1,figsize=(120,120))
plt.savefig('example.png', dpi=1000)
```

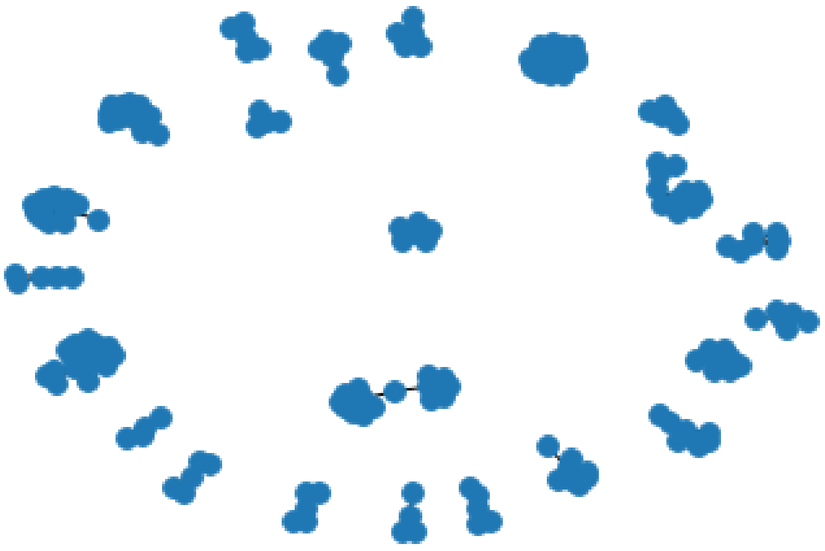
```
NETWORK SIZE
============
The network has 174 nodes and 154 edges

PENDANTS
============
The network has 143 pendants

DENSITY
============
The network density is 0.010231878280512923

DEGREE CENTRALITY
=================
The mean degree centrality is 0.010231878280512965, with stdev
0.011945260908062486
The maximum node is http://www.wikidata.org/entity/Q5373427, with value
0.07514450867052022
The minimum node is wcd_00153_id, with value 0.005780346820809248
(0.07514450867052022, 0.06358381502890173, 0.057803468208092484,
0.046242774566473986, 0.04046242774566474, 0.03468208092485549,
0.028901734104046242, 0.023121387283236993, 0.017341040462427744,
0.011560693641618497, 0.005780346820809248) (1, 2, 1, 2, 4, 2, 2, 8, 4, 5, 143)
```



```
EIGENVECTOR CENTRALITY
======================
The mean network eigenvector centrality is 0.01948514200092661, with stdev
0.07347435901965672
The maximum node is http://www.wikidata.org/entity/Q1382113, with value
0.5877661662137675
The minimum node is http://www.wikidata.org/entity/Q850141, with value
-4.505481786806643e-16
(0.5877661662137675, 0.4744595027388193, 0.26884388627369094,
0.2688438862736909, 0.1487606117642697, 0.14876061176426966,
0.14876061176426963, 0.1487606117642696, 0.12008327450942122,
0.12008327450942116, 0.12008327450942113, 1.3593413091090835e-16,
9.989522336346594e-17, 8.834850543356192e-17, 8.367196838271632e-17,
8.130062294824438e-17, 8.109920814682827e-17, 7.639250940834377e-17,
6.60755003837558e-17, 6.458331586029222e-17, 6.378391273135149e-17,
5.776875464768082e-17, 5.5230781654597724e-17, 4.632201986965634e-17,
4.5948913688461446e-17, 4.5729664583611263e-17, 4.180583512389912e-17,
3.990563383927098e-17, 3.933785144010534e-17, 3.796689971720141e-17,
3.729655473350136e-17, 3.518282345835072e-17, 3.3858485731042385e-17,
3.116169039624658e-17, 3.0462068968057656e-17, 2.96051565728719e-17,
2.928765390998258e-17, 2.8328372810121837e-17, 2.824097600371789e-17,
2.490232387743002e-17, 2.471227843456779e-17, 2.397858460927947e-17,
2.3693851342977602e-17, 2.353385305828489e-17, 1.96280254858043e-17,
1.9396502611160725e-17, 1.7312280663938313e-17, 1.706705487415864e-17,
1.677162700055526e-17, 1.6009894424159424e-17, 1.3715157481941634e-17,
1.3392917709502256e-17, 1.3282265459074696e-17, 1.2536087619363648e-17,
1.1003381032830206e-17, 1.0367419703382782e-17, 9.690003206518953e-18,
9.119550720730226e-18, 6.669266561467615e-18, 6.2847727103900965e-18,
5.925975685261885e-18, 5.6400307279347755e-18, 4.470552530857102e-18,
4.361772042560186e-18, 3.885819732618177e-18, 2.5991871601432675e-18,
2.293730052186802e-18, 2.216763629558276e-18, 1.5562080570519094e-18,
1.5101135079016543e-18, 8.735088056917535e-19, 5.091946402563726e-19,
-7.380313762569938e-20, -9.145440149184252e-20, -1.6131610381332627e-18,
-2.4637680242200984e-18, -3.6947795051584144e-18, -4.460444437411965e-18,
-4.9426428822043514e-18, -5.0168305851493625e-18, -5.9902170029824135e-18,
-6.478455626071773e-18, -7.657928674497709e-18, -7.796840606083429e-18,
-8.172173875109491e-18, -8.93151464013122e-18, -8.983016759598348e-18,
-1.0032161946479602e-17, -1.0104496019381209e-17, -1.0123396919182274e-17,
-1.1198386672833206e-17, -1.3557292598436024e-17, -1.3730489496385865e-17,
-1.4712533516235855e-17, -1.521153625256868e-17, -1.5629764138743687e-17,
-1.700655190433631e-17, -1.7044920801100344e-17, -1.7493966488533344e-17,
-1.770710757967251e-17, -1.8165666689094258e-17, -1.8892770988936693e-17,
-1.9027407083393462e-17, -2.009564437759557e-17, -2.010810736824598e-17,
-2.0648444074975188e-17, -2.082250898753491e-17, -2.141871505865804e-17,
-2.1858624313601425e-17, -2.273348040287359e-17, -2.2919382920802238e-17,
-2.4286128663675305e-17, -2.4291955939488523e-17, -2.437511230477948e-17,
-2.4619580104228264e-17, -2.5046226881248077e-17, -2.5087553268786578e-17,
-2.5717486384176555e-17, -2.7081292267934938e-17, -3.0491495862458354e-17,
-3.068317981835574e-17, -3.093754708903187e-17, -3.157642045569366e-17,
-3.2272613405298885e-17, -3.3422726711265095e-17, -3.511117111835248e-17,
-3.639011172442013e-17, -3.681622297742476e-17, -3.744625937459483e-17,
-3.7640692960354386e-17, -3.8087987972194085e-17, -3.8272236992131277e-17,
-3.881750104834077e-17, -4.0501962532597664e-17, -4.318125372162919e-17,
-4.3849139175554844e-17, -4.6004154690447276e-17, -4.8885964191348045e-17,
-4.946521968011863e-17, -5.0002431772889196e-17, -5.146258674314064e-17,
-5.800655455856678e-17, -5.925918849422934e-17, -6.154766862396167e-17,
-6.40759874005501e-17, -6.5617533442163e-17, -6.94373540330799e-17,
-7.191715408032303e-17, -7.285838599102591e-17, -7.700483730548221e-17,
-8.117098645054182e-17, -8.185599975814986e-17, -8.879302744879469e-17,
-9.083873672066412e-17, -9.446608991054129e-17, -9.852745480651999e-17,
-9.887923813067803e-17, -1.1231135809944714e-16, -1.1382903973526863e-16,
-1.162264728904461e-16, -1.2143064331837652e-16, -1.290582547627127e-16,
-1.2918140103891835e-16, -1.3250093073348863e-16, -1.446014962661383e-16,
-1.4969791370238877e-16, -1.5439038936193586e-16, -2.0566414218240156e-16,
-4.505481786806643e-16) (1, 1, 1, 2, 1, 3, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
```



```
CONNECTED COMPONENTS
====================
The graph has 23 connected components
Connected component 0 has 16 nodes
Connected component 1 has 14 nodes
Connected component 2 has 5 nodes
Connected component 3 has 15 nodes
Connected component 4 has 5 nodes
Connected component 5 has 12 nodes
Connected component 6 has 11 nodes
Connected component 7 has 4 nodes
Connected component 8 has 5 nodes
Connected component 9 has 8 nodes
Connected component 10 has 4 nodes
Connected component 11 has 8 nodes
Connected component 12 has 7 nodes
Connected component 13 has 5 nodes
Connected component 14 has 11 nodes
Connected component 15 has 8 nodes
Connected component 16 has 5 nodes
Connected component 17 has 6 nodes
Connected component 18 has 5 nodes
Connected component 19 has 5 nodes
Connected component 20 has 4 nodes
Connected component 21 has 5 nodes
Connected component 22 has 6 nodes

CLUSTERS
========
The graph has 174 clusters
Cluster 0 has 39 nodes
Cluster 1 has 12 nodes
Cluster 2 has 39 nodes
Cluster 3 has 40 nodes
Cluster 4 has 12 nodes
Cluster 5 has 40 nodes
Cluster 6 has 12 nodes
Cluster 7 has 38 nodes
Cluster 8 has 39 nodes
Cluster 9 has 40 nodes
Cluster 10 has 12 nodes
Cluster 11 has 38 nodes
Cluster 12 has 25 nodes
Cluster 13 has 38 nodes
Cluster 14 has 39 nodes
Cluster 15 has 40 nodes
Cluster 16 has 40 nodes
Cluster 17 has 39 nodes
Cluster 18 has 25 nodes
Cluster 19 has 25 nodes
Cluster 20 has 38 nodes
Cluster 21 has 12 nodes
Cluster 22 has 40 nodes
Cluster 23 has 40 nodes
Cluster 24 has 40 nodes
Cluster 25 has 25 nodes
Cluster 26 has 39 nodes
Cluster 27 has 39 nodes
Cluster 28 has 12 nodes
Cluster 29 has 37 nodes
Cluster 30 has 40 nodes
Cluster 31 has 25 nodes
Cluster 32 has 39 nodes
Cluster 33 has 25 nodes
Cluster 34 has 25 nodes
Cluster 35 has 38 nodes
Cluster 36 has 12 nodes
Cluster 37 has 12 nodes
Cluster 38 has 38 nodes
Cluster 39 has 40 nodes
Cluster 40 has 40 nodes
Cluster 41 has 39 nodes
Cluster 42 has 40 nodes
Cluster 43 has 39 nodes
Cluster 44 has 12 nodes
Cluster 45 has 38 nodes
Cluster 46 has 25 nodes
Cluster 47 has 25 nodes
Cluster 48 has 25 nodes
Cluster 49 has 25 nodes
Cluster 50 has 12 nodes
Cluster 51 has 25 nodes
Cluster 52 has 40 nodes
Cluster 53 has 25 nodes
Cluster 54 has 25 nodes
Cluster 55 has 25 nodes
Cluster 56 has 38 nodes
Cluster 57 has 40 nodes
Cluster 58 has 39 nodes
Cluster 59 has 12 nodes
Cluster 60 has 39 nodes
Cluster 61 has 25 nodes
Cluster 62 has 39 nodes
Cluster 63 has 12 nodes
Cluster 64 has 39 nodes
Cluster 65 has 25 nodes
Cluster 66 has 40 nodes
Cluster 67 has 38 nodes
Cluster 68 has 12 nodes
Cluster 69 has 25 nodes
Cluster 70 has 39 nodes
Cluster 71 has 39 nodes
Cluster 72 has 40 nodes
Cluster 73 has 40 nodes
Cluster 74 has 25 nodes
Cluster 75 has 25 nodes
Cluster 76 has 40 nodes
Cluster 77 has 12 nodes
Cluster 78 has 25 nodes
Cluster 79 has 25 nodes
Cluster 80 has 39 nodes
Cluster 81 has 39 nodes
Cluster 82 has 25 nodes
Cluster 83 has 40 nodes
Cluster 84 has 25 nodes
Cluster 85 has 39 nodes
Cluster 86 has 25 nodes
Cluster 87 has 25 nodes
Cluster 88 has 39 nodes
Cluster 89 has 25 nodes
Cluster 90 has 25 nodes
Cluster 91 has 25 nodes
Cluster 92 has 39 nodes
Cluster 93 has 40 nodes
Cluster 94 has 25 nodes
Cluster 95 has 39 nodes
Cluster 96 has 39 nodes
Cluster 97 has 12 nodes
Cluster 98 has 25 nodes
Cluster 99 has 25 nodes
Cluster 100 has 25 nodes
Cluster 101 has 39 nodes
Cluster 102 has 39 nodes
Cluster 103 has 12 nodes
Cluster 104 has 25 nodes
Cluster 105 has 25 nodes
Cluster 106 has 25 nodes
Cluster 107 has 25 nodes
Cluster 108 has 38 nodes
Cluster 109 has 40 nodes
Cluster 110 has 40 nodes
Cluster 111 has 25 nodes
Cluster 112 has 39 nodes
Cluster 113 has 40 nodes
Cluster 114 has 40 nodes
Cluster 115 has 25 nodes
Cluster 116 has 12 nodes
Cluster 117 has 25 nodes
Cluster 118 has 39 nodes
Cluster 119 has 39 nodes
Cluster 120 has 40 nodes
Cluster 121 has 39 nodes
Cluster 122 has 25 nodes
Cluster 123 has 12 nodes
Cluster 124 has 40 nodes
Cluster 125 has 25 nodes
Cluster 126 has 40 nodes
Cluster 127 has 39 nodes
Cluster 128 has 40 nodes
Cluster 129 has 40 nodes
Cluster 130 has 39 nodes
Cluster 131 has 25 nodes
Cluster 132 has 25 nodes
Cluster 133 has 12 nodes
Cluster 134 has 38 nodes
Cluster 135 has 25 nodes
Cluster 136 has 39 nodes
Cluster 137 has 25 nodes
Cluster 138 has 12 nodes
Cluster 139 has 25 nodes
Cluster 140 has 25 nodes
Cluster 141 has 25 nodes
Cluster 142 has 25 nodes
Cluster 143 has 25 nodes
Cluster 144 has 12 nodes
Cluster 145 has 25 nodes
Cluster 146 has 25 nodes
Cluster 147 has 40 nodes
Cluster 148 has 39 nodes
Cluster 149 has 25 nodes
Cluster 150 has 12 nodes
Cluster 151 has 12 nodes
Cluster 152 has 25 nodes
Cluster 153 has 25 nodes
Cluster 154 has 25 nodes
Cluster 155 has 25 nodes
Cluster 156 has 40 nodes
Cluster 157 has 25 nodes
Cluster 158 has 39 nodes
Cluster 159 has 25 nodes
Cluster 160 has 39 nodes
Cluster 161 has 39 nodes
Cluster 162 has 25 nodes
Cluster 163 has 12 nodes
Cluster 164 has 25 nodes
Cluster 165 has 40 nodes
Cluster 166 has 25 nodes
Cluster 167 has 38 nodes
Cluster 168 has 12 nodes
Cluster 169 has 25 nodes
Cluster 170 has 12 nodes
Cluster 171 has 40 nodes
Cluster 172 has 40 nodes
Cluster 173 has 39 nodes

Visualizing the graph:
```



B. 19th century German speaking female editors and their kinship relations as present in Wikidata

May 15, 2020

1 Network Analysis of RDF Graphs

In this notebook we provide basic facilities for performing network analyses of RDF graphs easily with Python rdflib and networkx

We do this in 4 steps: 1. Load an arbitrary RDF graph into rdflib 2. Get a subgraph of relevance (optional) 3. Convert the rdflib Graph into an networkx Graph, as shown here 4. Get an network analysis report by running networkx's algorithms on that data structure

[13]:

1.1 0. Preparation

```
# Install required packages in the current Jupyter kernel
# Uncomment the following lines if you need to install these libraries
# If you run into permission issues, try with the --user option
import sys
# !pip install -q rdflib networkx matplotlib scipy
!{sys.executable} -m pip install rdflib networkx matplotlib scipy --user

# Imports
from rdflib import Graph as RDFGraph
from rdflib.extras.external_graph_libs import rdflib_to_networkx_graph
import networkx as nx
from networkx import Graph as NXGraph
import matplotlib.pyplot as plt
import statistics
import collections
```

```
Requirement already satisfied: rdflib in /home/amp/.local/lib/python3.8/site-
packages (5.0.0)
Requirement already satisfied: networkx in /home/amp/.local/lib/python3.8/site-
packages (2.4)
Requirement already satisfied: matplotlib in
/home/amp/.local/lib/python3.8/site-packages (3.2.1)
Requirement already satisfied: scipy in /home/amp/.local/lib/python3.8/site-
packages (1.4.1)
Requirement already satisfied: pyparsing in /usr/lib/python3/dist-packages (from
rdflib) (2.4.6)
Requirement already satisfied: six in /usr/lib/python3/dist-packages (from rdflib)
(1.14.0)
```

```
Requirement already satisfied: isodate in /home/amp/.local/lib/python3.8/site-
packages (from rdflib) (0.6.0)
Requirement already satisfied: decorator>=4.3.0 in /usr/lib/python3/dist-
packages (from networkx) (4.4.2)
Requirement already satisfied: kiwisolver>=1.0.1 in
/home/amp/.local/lib/python3.8/site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: numpy>=1.11 in /usr/lib/python3/dist-packages
(from matplotlib) (1.17.4)
Requirement already satisfied: cycler>=0.10 in
/home/amp/.local/lib/python3.8/site-packages (from matplotlib) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/lib/python3/dist-
packages (from matplotlib) (2.7.3)
```

1.2 1. Loading RDF

The first thing to do is to load the RDF graph we want to perform the network analysis on. By executing the next cell, we'll be asked to fill in the path to an RDF graph. This can be any path, local or online, that we can look up.

Any of the Turtle (`ttl.`) files that we include with this notebook will do; for example, `bsbm-sample.ttl`. But any Web location that leads to an RDF file (for example, the GitHub copy of that same file at https://raw.githubusercontent.com/albertmeronyo/rdf-network-analysis/master/bsbm-sample.ttl; or any other RDF file on the Web like https://raw.githubusercontent.com/albertmeronyo/lodapi/master/ghostbusters.ttl) will work too.

[18]:
```
# RDF graph loading
path = input("Path or URI of the RDF graph to load: ")
rg = RDFGraph()
rg.parse(path, format='turtle')
print("rdflib Graph loaded successfully with {} triples".format(len(rg)))
```

```
Path or URI of the RDF graph to load: wechanged-german.ttl
rdflib Graph loaded successfully with 53 triples
```

1.3 2. Get a subgraph out of the loaded RDF graph (optional)

This cell can be skipped altogether without affecting the rest of the notebook; but it will be useful if instead of using the whole RDF grahp of the previous step, we just want to use a subgraph that's included in it.

By executing the next cell, we'll be asked two things:

- The URI of the ''entiy" type we are interested in (e.g. `http://dbpedia.org/ontology/Band`)
- The URI of the ''relation" connecting entities we are interested in (e.g. `http://dbpedia.org/ontology/influencedBy`)

Using these two, the notebook will replace the original graph with the subgraph that's constructed by those entity types and relations only.

[ ]:
```
# Subgraph construction (optional)
entity = input("Entity type to build nodes of the subgraph with: ")
relation = input("Relation type to build edges of the subgraph with: ")

# TODO: Use entity and relation as parameters of a CONSTRUCT query
query = """
PREFIX bsbm: <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/vocabulary/>
CONSTRUCT {{ ?u a {} . ?u {} ?v }} WHERE {{ ?u a {} . ?u {} ?v }}""".
 ↪format(entity, relation, entity, relation)
# print(query)
subg = rg.query(query)

rg = subg
```

1.4 3. Converting rdflib.Graph to networkx.Graph

Thanks to the great work done by the rdflib developers this step, which converts the basic graph data structure of rdflib into its equivalent in networkx, is straightforward. Just run the next cell to make our RDF dataset ready for network analysis!

[19]:
```
# Conversion of rdflib.Graph to networkx.Graph
G = rdflib_to_networkx_graph(rg)
print("networkx Graph loaded successfully with length {}".format(len(G)))
```

```
networkx Graph loaded successfully with length 65
```

1.5 4. Network analysis

At this point we can run the network analysis on our RDF graph by using the networkx algorithms. Exeucting the next cell will output a full network analysis report, with the following parts:

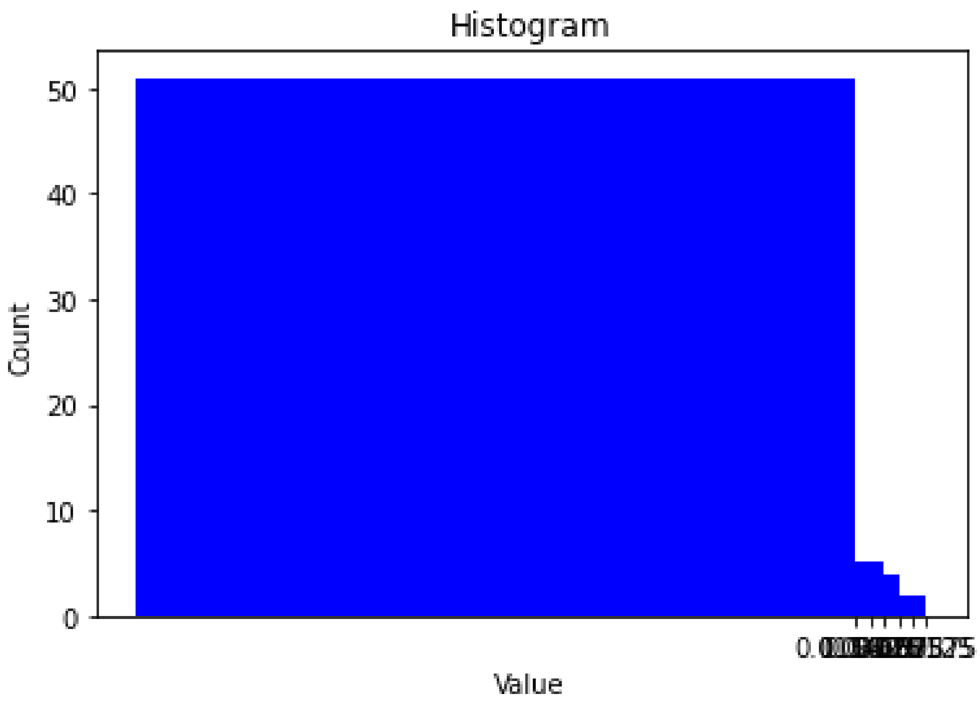
- General network metrics (network size, pendants, density)
- Node centrality metrics (degree, eigenvector, betwenness). For these, averages, stdevs, maximum, minimum and distribution histograms are given
- Clustering metrics (connected components, clustering)
- Overall network plot

The report can be easily selected and copy-pasted for further use in other tools.

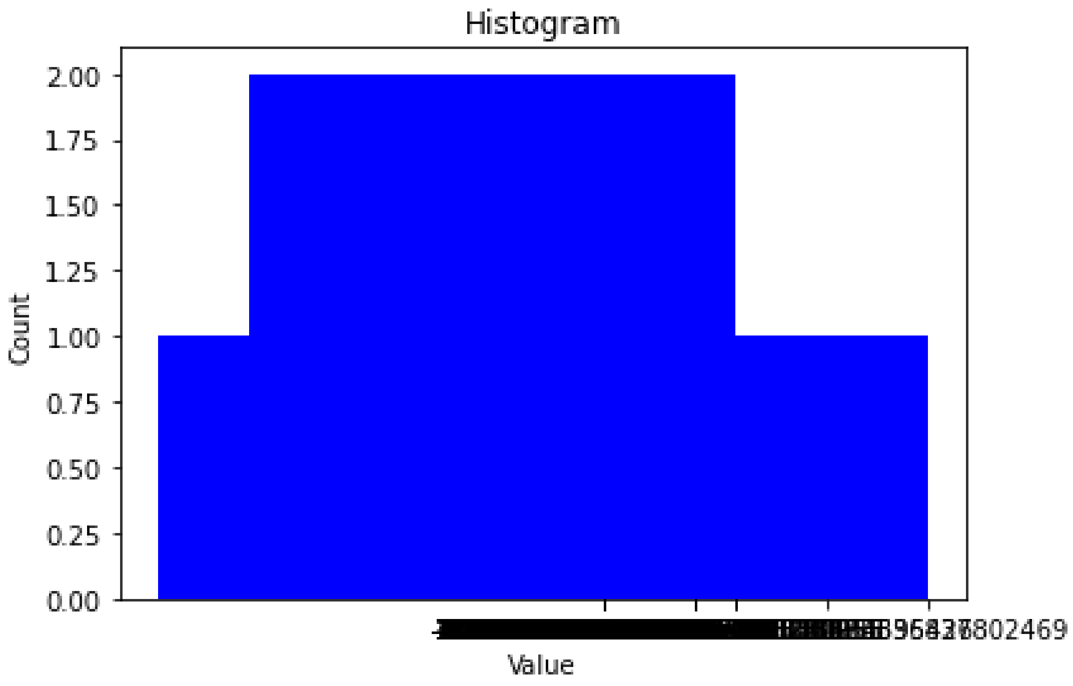
[20]:
```
# Analysis

def mean(numbers):
    return float(sum(numbers)) / max(len(numbers), 1)

def number_of_pendants(g):
    """
    Equals the number of nodes with degree 1
    """
    pendants = 0
    for u in g:
        if g.degree[u] == 1:
            pendants += 1
    return pendants


def histogram(l):
    degree_sequence = sorted([d for n, d in list(l.items())], reverse=True)
    degreeCount = collections.Counter(degree_sequence)
    deg, cnt = zip(*degreeCount.items())
    print(deg, cnt)

    fig, ax = plt.subplots()
    plt.bar(deg, cnt, width=0.80, color='b')

    plt.title("Histogram")
    plt.ylabel("Count")
    plt.xlabel("Value")
    ax.set_xticks([d + 0.4 for d in deg])
    ax.set_xticklabels(deg)

    plt.show()

# Network size
print("NETWORK SIZE")
print("============")
print("The network has {} nodes and {} edges".format(G.number_of_nodes(), G.
 ↪number_of_edges()))
print()

# Network size
print("PENDANTS")
print("============")
print("The network has {} pendants".format(number_of_pendants(G)))
print()

# Density
print("DENSITY")
print("============")
print("The network density is {}".format(nx.density(G)))
print()

# Degree centrality -- mean and stdev
dc = nx.degree_centrality(G)
degrees = []
for k,v in dc.items():
    degrees.append(v)
```

```
print("DEGREE CENTRALITY")
print("=================")
print("The mean degree centrality is {}, with stdev {}".format(mean(degrees),␣
 ↪statistics.stdev(degrees)))
print("The maximum node is {}, with value {}".format(max(dc, key=dc.get),␣
 ↪max(dc.values())))
print("The minimum node is {}, with value {}".format(min(dc, key=dc.get),␣
 ↪min(dc.values())))
histogram(dc)
print()

# Eigenvector centrality -- mean and stdev
ec = nx.eigenvector_centrality_numpy(G)
degrees = []
for k,v in ec.items():
    degrees.append(v)

print("EIGENVECTOR CENTRALITY")
print("======================")
print("The mean network eigenvector centrality is {}, with stdev {}".
 ↪format(mean(degrees), statistics.stdev(degrees)))
print("The maximum node is {}, with value {}".format(max(ec, key=ec.get),␣
 ↪max(ec.values())))
print("The minimum node is {}, with value {}".format(min(ec, key=ec.get),␣
 ↪min(ec.values())))
histogram(ec)
print()

# Betweenness centrality -- mean and stdev
# bc = nx.betweenness_centrality(G)
# degrees = []
# for k,v in bc.items():
#     degrees.append(v)
# print("BETWEENNESS CENTRALITY")
# print("======================")
# print("The mean betwenness centrality is {}, with stdev {}".
 ↪format(mean(degrees), statistics.stdev(degrees)))
# print("The maximum node is {}, with value {}".format(max(bc, key=bc.get),␣
 ↪max(bc.values())))
# print("The minimum node is {}, with value {}".format(min(bc, key=bc.get),␣
 ↪min(bc.values())))
# histogram(bc)
# print()
# Connected components
cc = list(nx.connected_components(G))
print("CONNECTED COMPONENTS")
print("====================")
print("The graph has {} connected components".format(len(cc)))
for i,c in enumerate(cc):
    print("Connected component {} has {} nodes".format(i,len(c)))
print()

# Clusters
cl = nx.clustering(G)
print("CLUSTERS")
print("========")
print("The graph has {} clusters".format(len(cl)))
for i,c in enumerate(cl):
    print("Cluster {} has {} nodes".format(i,len(c)))
print()

# Plot
print("Visualizing the graph:")
plt.plot()
plt.figure(1)
nx.draw(G, with_labels=False, font_weight='normal', node_size=60, font_size=8)
plt.figure(1,figsize=(120,120))
plt.savefig('example.png', dpi=1000)
```

```
NETWORK SIZE
============
The network has 65 nodes and 53 edges

PENDANTS
============
The network has 51 pendants

DENSITY
============
The network density is 0.02548076923076923

DEGREE CENTRALITY
=================
The mean degree centrality is 0.02548076923076923, with stdev
0.020774719730186218
The maximum node is http://www.wikidata.org/entity/Q165824, with value 0.09375
The minimum node is wcd_00814_id, with value 0.015625
(0.09375, 0.078125, 0.0625, 0.046875, 0.03125, 0.015625) (2, 2, 4, 5, 1, 51)
```

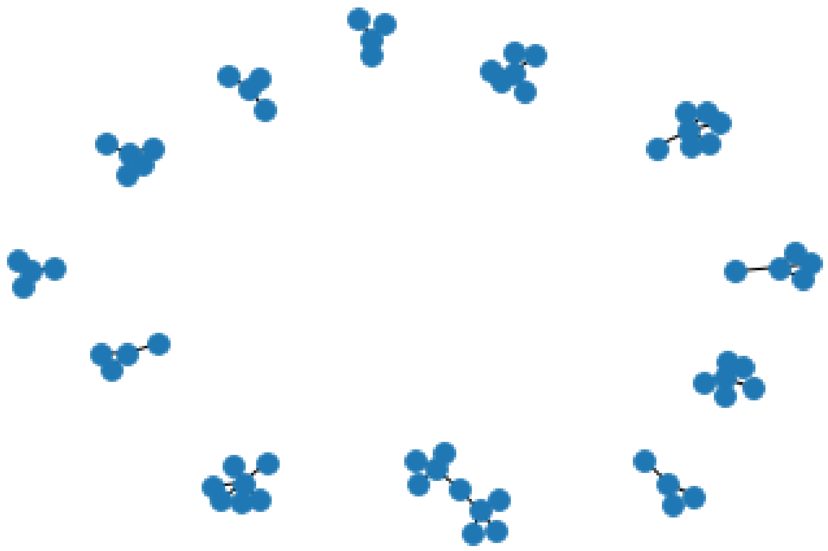


```
EIGENVECTOR CENTRALITY
======================
The mean network eigenvector centrality is 0.052190328754421186, with stdev
0.11339581003839311
The maximum node is http://www.wikidata.org/entity/Q165824, with value
0.5823336837802469
The minimum node is http://www.wikidata.org/entity/Q2653682, with value
-4.221865487293616e-16
(0.5823336837802469, 0.40110781684595426, 0.23773673088280958,
0.23773673088280955, 0.23773673088280953, 0.2377367308828095,
0.16375158051905314, 0.1637515805190531, 0.16375158051905309,
0.16375158051905306, 0.16375158051905303, 4.0375682011403296e-16,
3.867620361637153e-16, 2.423140802377901e-16, 2.1493997183573874e-16,
2.0826656295842276e-16, 1.8925354328681477e-16, 1.860937486981313e-16,
1.7617115305582745e-16, 1.667799235809163e-16, 1.5837406027033936e-16,
1.3001507409184495e-16, 1.2555668835368535e-16, 1.1354908529513395e-16,
1.0195879145351594e-16, 9.659051261599858e-17, 8.249547678468506e-17,
7.150789936020287e-17, 6.477197106861316e-17, 6.34748456895395e-17,
6.255159781101699e-17, 5.748386506242491e-17, 4.5070823959909377e-17,
4.028075437461217e-17, 3.191177899331292e-17, 1.7134628208983114e-17,
1.504830421598233e-17, 1.5222146334878828e-18, -1.2422309969230075e-18,
-8.56840964600123e-18, -1.0566991786028656e-17, -1.3380020371347866e-17,
-1.82290962330364e-17, -2.5459354322188953e-17, -2.9381498680853597e-17,
-5.607146449104495e-17, -6.208476377865393e-17, -6.278772145308986e-17,
-1.0752076302444307e-16, -1.0828082789917711e-16, -1.7349787989201016e-16,
-1.765445338168193e-16, -1.879654759105251e-16, -1.9130239507871805e-16,
-1.93439031608548e-16, -1.9861678449786301e-16, -1.9935548922841931e-16,
-2.3141199604139336e-16, -2.477318548940688e-16, -2.722433427921724e-16,
-3.7706856978891896e-16, -4.221865487293616e-16) (1, 1, 1, 1, 2, 2, 1, 1, 1, 2,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
```



```
CONNECTED COMPONENTS
====================
The graph has 12 connected components
Connected component 0 has 5 nodes
Connected component 1 has 7 nodes
Connected component 2 has 5 nodes
Connected component 3 has 4 nodes
Connected component 4 has 9 nodes
Connected component 5 has 4 nodes
Connected component 6 has 6 nodes
Connected component 7 has 6 nodes
Connected component 8 has 4 nodes
Connected component 9 has 7 nodes
Connected component 10 has 4 nodes
Connected component 11 has 4 nodes
```

```
CLUSTERS
========
The graph has 65 clusters
Cluster 0 has 37 nodes
Cluster 1 has 12 nodes
Cluster 2 has 38 nodes
Cluster 3 has 37 nodes
Cluster 4 has 37 nodes
Cluster 5 has 25 nodes
Cluster 6 has 37 nodes
Cluster 7 has 25 nodes
Cluster 8 has 39 nodes
Cluster 9 has 40 nodes
Cluster 10 has 39 nodes
Cluster 11 has 25 nodes
Cluster 12 has 25 nodes
Cluster 13 has 37 nodes
Cluster 14 has 25 nodes
Cluster 15 has 25 nodes
Cluster 16 has 39 nodes
Cluster 17 has 25 nodes
Cluster 18 has 25 nodes
Cluster 19 has 39 nodes
Cluster 20 has 12 nodes
Cluster 21 has 37 nodes
Cluster 22 has 12 nodes
Cluster 23 has 12 nodes
Cluster 24 has 25 nodes
Cluster 25 has 37 nodes
Cluster 26 has 12 nodes
Cluster 27 has 25 nodes
Cluster 28 has 25 nodes
Cluster 29 has 37 nodes
Cluster 30 has 12 nodes
Cluster 31 has 25 nodes
Cluster 32 has 39 nodes
Cluster 33 has 12 nodes
Cluster 34 has 25 nodes
Cluster 35 has 40 nodes
Cluster 36 has 25 nodes
Cluster 37 has 12 nodes
Cluster 38 has 40 nodes
Cluster 39 has 12 nodes
Cluster 40 has 25 nodes
Cluster 41 has 39 nodes
Cluster 42 has 25 nodes
Cluster 43 has 38 nodes
Cluster 44 has 12 nodes
Cluster 45 has 25 nodes
Cluster 46 has 25 nodes
Cluster 47 has 25 nodes
Cluster 48 has 25 nodes
Cluster 49 has 25 nodes
Cluster 50 has 25 nodes
Cluster 51 has 37 nodes
Cluster 52 has 25 nodes
Cluster 53 has 12 nodes
Cluster 54 has 25 nodes
Cluster 55 has 40 nodes
Cluster 56 has 40 nodes
Cluster 57 has 40 nodes
Cluster 58 has 25 nodes
Cluster 59 has 12 nodes
Cluster 60 has 38 nodes
Cluster 61 has 25 nodes
Cluster 62 has 12 nodes
Cluster 63 has 25 nodes
Cluster 64 has 25 nodes

Visualizing the graph:
```



rdf-network-analysis

May 15, 2020

1 Network Analysis of RDF Graphs

In this notebook we provide basic facilities for performing network analyses of RDF graphs easily with Python rdflib and networkx

We do this in 4 steps: 1. Load an arbitrary RDF graph into rdflib 2. Get a subgraph of relevance (optional) 3. Convert the rdflib Graph into an networkx Graph, as shown here 4. Get an network analysis report by running networkx's algorithms on that data structure

1.1 0. Preparation

[13]:
```
# Install required packages in the current Jupyter kernel
# Uncomment the following lines if you need to install these libraries
# If you run into permission issues, try with the --user option
import sys
# !pip install -q rdflib networkx matplotlib scipy
!{sys.executable} -m pip install rdflib networkx matplotlib scipy --user

# Imports
from rdflib import Graph as RDFGraph
from rdflib.extras.external_graph_libs import rdflib_to_networkx_graph
import networkx as nx
from networkx import Graph as NXGraph
import matplotlib.pyplot as plt
import statistics
import collections
```

```
Requirement already satisfied: rdflib in /home/amp/.local/lib/python3.8/site-
packages (5.0.0)
Requirement already satisfied: networkx in /home/amp/.local/lib/python3.8/site-
packages (2.4)
Requirement already satisfied: matplotlib in
/home/amp/.local/lib/python3.8/site-packages (3.2.1)
Requirement already satisfied: scipy in /home/amp/.local/lib/python3.8/site-
packages (1.4.1)
Requirement already satisfied: pyparsing in /usr/lib/python3/dist-packages (from
rdflib) (2.4.6)
Requirement already satisfied: six in /usr/lib/python3/dist-packages (from
rdflib) (1.14.0)
```

```
Requirement already satisfied: isodate in /home/amp/.local/lib/python3.8/site-
packages (from rdflib) (0.6.0)
Requirement already satisfied: decorator>=4.3.0 in /usr/lib/python3/dist-
packages (from networkx) (4.4.2)
Requirement already satisfied: kiwisolver>=1.0.1 in
/home/amp/.local/lib/python3.8/site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: numpy>=1.11 in /usr/lib/python3/dist-packages
(from matplotlib) (1.17.4)
Requirement already satisfied: cycler>=0.10 in
/home/amp/.local/lib/python3.8/site-packages (from matplotlib) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/lib/python3/dist-
packages (from matplotlib) (2.7.3)
```

1.2 1. Loading RDF

The first thing to do is to load the RDF graph we want to perform the network analysis on. By executing the next cell, we'll be asked to fill in the path to an RDF graph. This can be any path, local or online, that we can look up.

Any of the Turtle (`ttl.`) files that we include with this notebook will do; for example, `bsbm-sample.ttl`. But any Web location that leads to an RDF file (for example, the GitHub copy of that same file at https://raw.githubusercontent.com/albertmeronyo/rdf-network-analysis/master/bsbm-sample.ttl; or any other RDF file on the Web like https://raw.githubusercontent.com/albertmeronyo/lodapi/master/ghostbusters.ttl) will work too.

[14]:
```python
# RDF graph loading
path = input("Path or URI of the RDF graph to load: ")
rg = RDFGraph()
rg.parse(path, format='turtle')
print("rdflib Graph loaded successfully with {} triples".format(len(rg)))
```

```
Path or URI of the RDF graph to load: wechanged-british.ttl
rdflib Graph loaded successfully with 155 triples
```

1.3 2. Get a subgraph out of the loaded RDF graph (optional)

This cell can be skipped altogether without affecting the rest of the notebook; but it will be useful if instead of using the whole RDF grahp of the previous step, we just want to use a subgraph that's included in it.

By executing the next cell, we'll be asked two things:

- The URI of the ''entiy'' type we are interested in (e.g. `http://dbpedia.org/ontology/Band`)
- The URI of the ''relation'' connecting entities we are interested in (e.g. `http://dbpedia.org/ontology/influencedBy`)

Using these two, the notebook will replace the original graph with the subgraph that's constructed by those entity types and relations only.

[ ]:
```
# Subgraph construction (optional)
entity = input("Entity type to build nodes of the subgraph with: ")
relation = input("Relation type to build edges of the subgraph with: ")

# TODO: Use entity and relation as parameters of a CONSTRUCT query
query = """
PREFIX bsbm: <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/vocabulary/>
CONSTRUCT {{ ?u a {} . ?u {} ?v }} WHERE {{ ?u a {} . ?u {} ?v }}""".
↪format(entity, relation, entity, relation)
# print(query)
subg = rg.query(query)

rg = subg
```

1.4 3. Converting rdflib.Graph to networkx.Graph

Thanks to the great work done by the rdflib developers this step, which converts the basic graph data structure of rdflib into its equivalent in networkx, is straightforward. Just run the next cell to make our RDF dataset ready for network analysis!

[15]:
```
# Conversion of rdflib.Graph to networkx.Graph
G = rdflib_to_networkx_graph(rg)
print("networkx Graph loaded successfully with length {}".format(len(G)))
```

```
networkx Graph loaded successfully with length 174
```

1.5 4. Network analysis

At this point we can run the network analysis on our RDF graph by using the networkx algorithms. Exeucting the next cell will output a full network analysis report, with the following parts:

- General network metrics (network size, pendants, density)
- Node centrality metrics (degree, eigenvector, betwenness). For these, averages, stdevs, maximum, minimum and distribution histograms are given
- Clustering metrics (connected components, clustering)
- Overall network plot

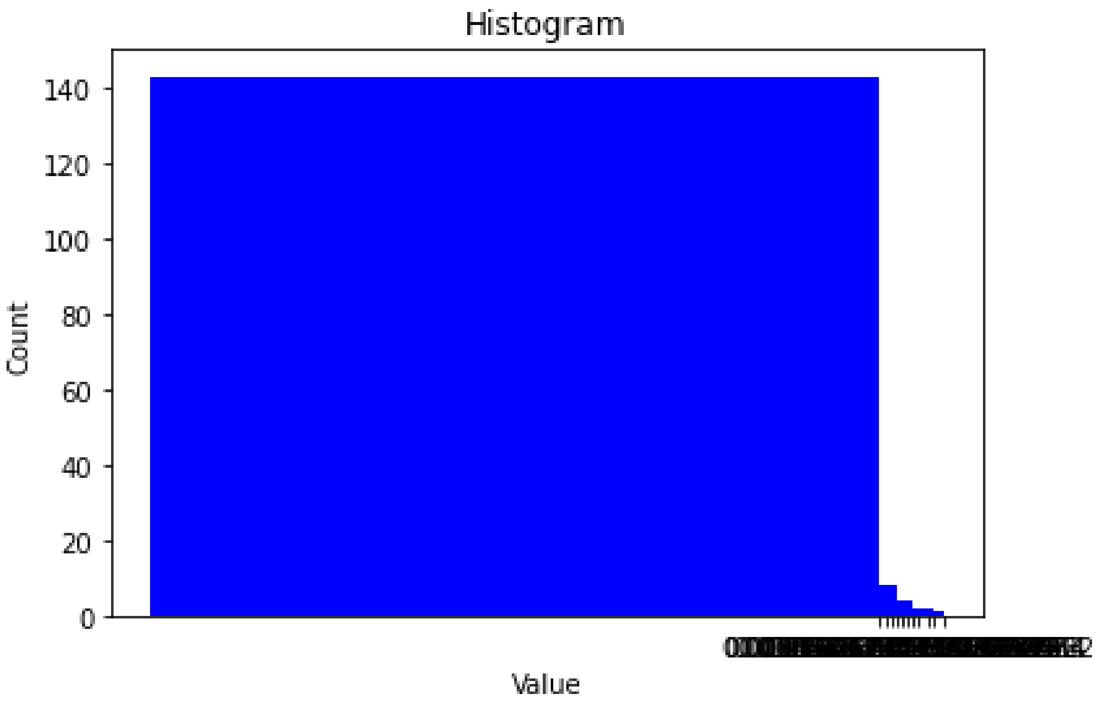
The report can be easily selected and copy-pasted for further use in other tools.

[17]:
```
# Analysis

def mean(numbers):
    return float(sum(numbers)) / max(len(numbers), 1)

def number_of_pendants(g):
    """
    Equals the number of nodes with degree 1
    """
    pendants = 0
    for u in g:
        if g.degree[u] == 1:
            pendants += 1
    return pendants


def histogram(l):
    degree_sequence = sorted([d for n, d in list(l.items())], reverse=True)
    degreeCount = collections.Counter(degree_sequence)
    deg, cnt = zip(*degreeCount.items())
    print(deg, cnt)

    fig, ax = plt.subplots()
    plt.bar(deg, cnt, width=0.80, color='b')

    plt.title("Histogram")
    plt.ylabel("Count")
    plt.xlabel("Value")
    ax.set_xticks([d + 0.4 for d in deg])
    ax.set_xticklabels(deg)

    plt.show()

# Network size
print("NETWORK SIZE")
print("============")
print("The network has {} nodes and {} edges".format(G.number_of_nodes(), G.
 ↪number_of_edges()))
print()

# Network size
print("PENDANTS")
print("============")
print("The network has {} pendants".format(number_of_pendants(G)))
print()

# Density
print("DENSITY")
print("============")
print("The network density is {}".format(nx.density(G)))
print()

# Degree centrality -- mean and stdev
dc = nx.degree_centrality(G)
degrees = []
for k,v in dc.items():
    degrees.append(v)
```

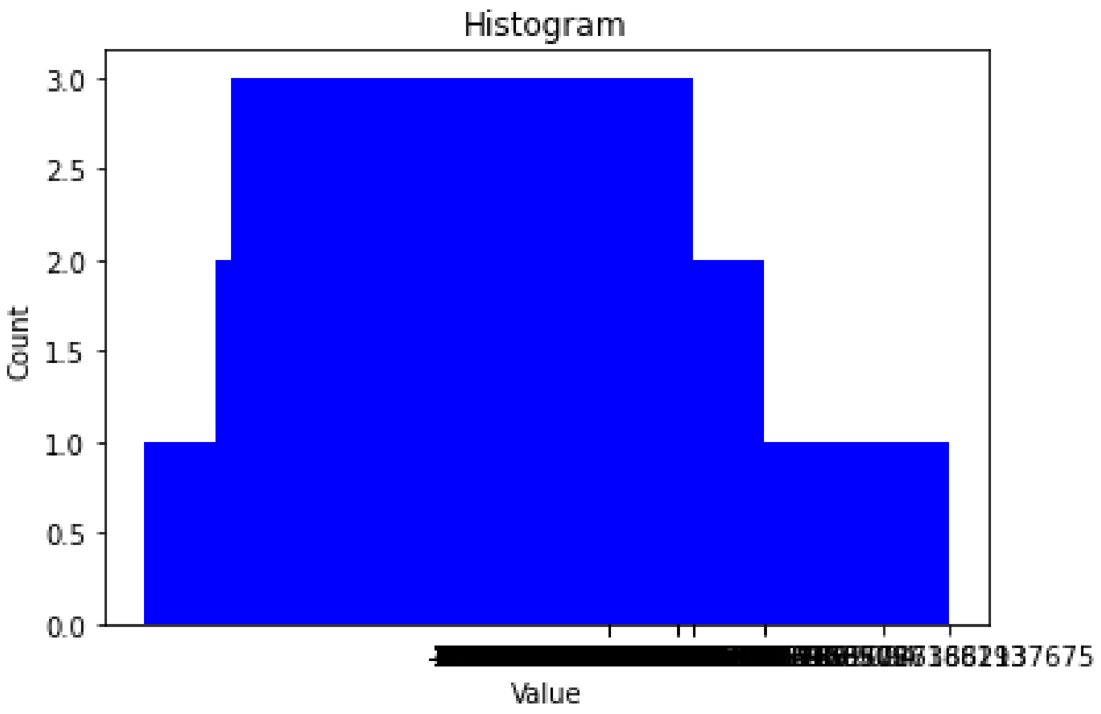
```
print("DEGREE CENTRALITY")
print("=================")
print("The mean degree centrality is {}, with stdev {}".format(mean(degrees),␣
↪statistics.stdev(degrees)))
print("The maximum node is {}, with value {}".format(max(dc, key=dc.get),␣
↪max(dc.values())))
print("The minimum node is {}, with value {}".format(min(dc, key=dc.get),␣
↪min(dc.values())))
histogram(dc)
print()

# Eigenvector centrality -- mean and stdev
ec = nx.eigenvector_centrality_numpy(G)
degrees = []
for k,v in ec.items():
    degrees.append(v)

print("EIGENVECTOR CENTRALITY")
print("======================")
print("The mean network eigenvector centrality is {}, with stdev {}".
↪format(mean(degrees), statistics.stdev(degrees)))
print("The maximum node is {}, with value {}".format(max(ec, key=ec.get),␣
↪max(ec.values())))
print("The minimum node is {}, with value {}".format(min(ec, key=ec.get),␣
↪min(ec.values())))
histogram(ec)
print()

# Betweenness centrality -- mean and stdev
# bc = nx.betweenness_centrality(G)
# degrees = []
# for k,v in bc.items():
#     degrees.append(v)
# print("BETWEENNESS CENTRALITY")
# print("======================")
# print("The mean betwenness centrality is {}, with stdev {}".
↪format(mean(degrees), statistics.stdev(degrees)))
# print("The maximum node is {}, with value {}".format(max(bc, key=bc.get),␣
↪max(bc.values())))
# print("The minimum node is {}, with value {}".format(min(bc, key=bc.get),␣
↪min(bc.values())))
# histogram(bc)
# print()
```

```python
# Connected components
cc = list(nx.connected_components(G))
print("CONNECTED COMPONENTS")
print("====================")
print("The graph has {} connected components".format(len(cc)))
for i,c in enumerate(cc):
    print("Connected component {} has {} nodes".format(i,len(c)))
print()

# Clusters
cl = nx.clustering(G)
print("CLUSTERS")
print("========")
print("The graph has {} clusters".format(len(cl)))
for i,c in enumerate(cl):
    print("Cluster {} has {} nodes".format(i,len(c)))
print()

# Plot
print("Visualizing the graph:")
plt.plot()
plt.figure(1)
nx.draw(G, with_labels=False, font_weight='normal', node_size=60, font_size=8)
plt.figure(1,figsize=(120,120))
plt.savefig('example.png', dpi=1000)
```

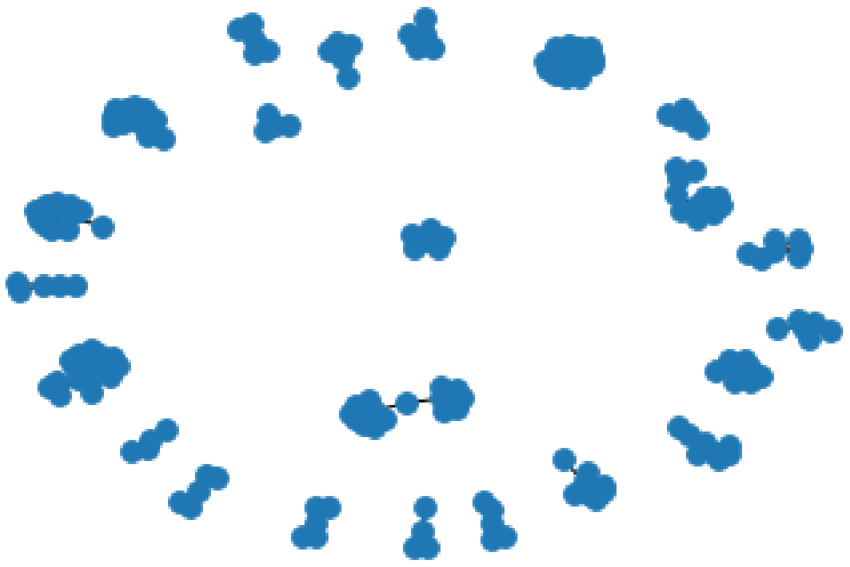
```
NETWORK SIZE
============
The network has 174 nodes and 154 edges

PENDANTS
============
The network has 143 pendants

DENSITY
============
The network density is 0.010231878280512923

DEGREE CENTRALITY
=================
The mean degree centrality is 0.010231878280512965, with stdev
0.011945260908062486
The maximum node is http://www.wikidata.org/entity/Q5373427, with value
0.07514450867052022
The minimum node is wcd_00153_id, with value 0.005780346820809248
(0.07514450867052022, 0.06358381502890173, 0.057803468208092484,
0.046242774566473986, 0.04046242774566474, 0.03468208092485549,
0.028901734104046242, 0.023121387283236993, 0.017341040462427744,
0.011560693641618497, 0.005780346820809248) (1, 2, 1, 2, 4, 2, 2, 8, 4, 5, 143)
```



```
EIGENVECTOR CENTRALITY
======================
The mean network eigenvector centrality is 0.01948514200092661, with stdev
0.07347435901965672
The maximum node is http://www.wikidata.org/entity/Q1382113, with value
0.5877661662137675
The minimum node is http://www.wikidata.org/entity/Q850141, with value
-4.505481786806643e-16
(0.5877661662137675, 0.4744595027388193, 0.26884388627369094,
0.2688438862736909, 0.1487606117642697, 0.14876061176426966,
0.14876061176426963, 0.1487606117642696, 0.12008327450942122,
0.12008327450942116, 0.12008327450942113, 1.3593413091090835e-16,
9.989522336346594e-17, 8.834850543356192e-17, 8.367196838271632e-17,
8.130062294824438e-17, 8.109920814682827e-17, 7.639250940834377e-17,
6.60755003837558e-17, 6.458331586029222e-17, 6.378391273135149e-17,
5.776875464768082e-17, 5.5230781654597724e-17, 4.632201986965634e-17,
4.5948913688461446e-17, 4.5729664583611263e-17, 4.180583512389912e-17,
3.990563383927098e-17, 3.933785144010534e-17, 3.796689971720141e-17,
3.729655473350136e-17, 3.518282345835072e-17, 3.3858485731042385e-17,
3.116169039624658e-17, 3.0462068968057656e-17, 2.96051565728719e-17,
2.928765390998258e-17, 2.8328372810121837e-17, 2.824097600371789e-17,
2.490232387743002e-17, 2.471227843456779e-17, 2.397858460927947e-17,
2.3693851342977602e-17, 2.353385305828489e-17, 1.96280254858043e-17,
1.9396502611160725e-17, 1.7312280663938313e-17, 1.706705487415864e-17,
1.677162700055526e-17, 1.6009894424159424e-17, 1.3715157481941634e-17,
1.3392917709502256e-17, 1.3282265459074696e-17, 1.2536087619363648e-17,
1.1003381032830206e-17, 1.0367419703382782e-17, 9.690003206518953e-18,
9.119550720730226e-18, 6.669266561467615e-18, 6.2847727103900965e-18,
5.925975685261885e-18, 5.6400307279347755e-18, 4.470552530857102e-18,
4.361772042560186e-18, 3.885819732618177e-18, 2.5991871601432675e-18,
2.293730052186802e-18, 2.216763629558276e-18, 1.5562080570519094e-18,
1.5101135079016543e-18, 8.735088056917535e-19, 5.091946402563726e-19,
-7.380313762569938e-20, -9.145440149184252e-20, -1.6131610381332627e-18,
-2.4637680242200984e-18, -3.6947795051584144e-18, -4.460444437411965e-18,
-4.9426428822043514e-18, -5.0168305851493625e-18, -5.9902170029824135e-18,
-6.478455626071773e-18, -7.657928674497709e-18, -7.796840606083429e-18,
-8.172173875109491e-18, -8.93151464013122e-18, -8.983016759598348e-18,
-1.0032161946479602e-17, -1.0104496019381209e-17, -1.0123396919182274e-17,
-1.1198386672833206e-17, -1.3557292598436024e-17, -1.3730489496385865e-17,
-1.4712533516235855e-17, -1.521153625256868e-17, -1.5629764138743687e-17,
-1.700655190433631e-17, -1.7044920801100344e-17, -1.7493966488533344e-17,
-1.770710757967251e-17, -1.8165666689094258e-17, -1.8892770988936693e-17,
-1.9027407083393462e-17, -2.009564437759557e-17, -2.010810736824598e-17,
-2.0648444074975188e-17, -2.082250898753491e-17, -2.141871505865804e-17,
-2.1858624313601425e-17, -2.273348040287359e-17, -2.2919382920802238e-17,
-2.4286128663675305e-17, -2.4291955939488523e-17, -2.437511230477948e-17,
-2.4619580104228264e-17, -2.5046226881248077e-17, -2.5087553268786578e-17,
-2.5717486384176555e-17, -2.7081292267934938e-17, -3.0491495862458354e-17,
-3.068317981835574e-17, -3.093754708903187e-17, -3.157642045569366e-17,
-3.2272613405298885e-17, -3.3422726711265095e-17, -3.511117111835248e-17,
-3.639011172442013e-17, -3.681622297742476e-17, -3.744625937459483e-17,
-3.7640692960354386e-17, -3.8087987972194085e-17, -3.8272236992131277e-17,
-3.881750104834077e-17, -4.0501962532597664e-17, -4.318125372162919e-17,
-4.3849139175554844e-17, -4.6004154690447276e-17, -4.8885964191348045e-17,
-4.946521968011863e-17, -5.0002431772889196e-17, -5.146258674314064e-17,
-5.800655455856678e-17, -5.925918849422934e-17, -6.154766862396167e-17,
-6.40759874005501e-17, -6.5617533442163e-17, -6.94373540330799e-17,
-7.191715408032303e-17, -7.285838599102591e-17, -7.700483730548221e-17,
-8.117098645054182e-17, -8.185599975814986e-17, -8.879302744879469e-17,
-9.083873672066412e-17, -9.446608991054129e-17, -9.852745480651999e-17,
-9.887923813067803e-17, -1.1231135809944714e-16, -1.1382903973526863e-16,
-1.162264728904461e-16, -1.2143064331837652e-16, -1.290582547627127e-16,
-1.2918140103891835e-16, -1.3250093073348863e-16, -1.446014962661383e-16,
-1.4969791370238877e-16, -1.5439038936193586e-16, -2.0566414218240156e-16,
-4.505481786806643e-16) (1, 1, 1, 2, 1, 3, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
```



```
CONNECTED COMPONENTS
====================
The graph has 23 connected components
Connected component 0 has 16 nodes
Connected component 1 has 14 nodes
Connected component 2 has 5 nodes
Connected component 3 has 15 nodes
Connected component 4 has 5 nodes
Connected component 5 has 12 nodes
Connected component 6 has 11 nodes
Connected component 7 has 4 nodes
Connected component 8 has 5 nodes
Connected component 9 has 8 nodes
Connected component 10 has 4 nodes
Connected component 11 has 8 nodes
Connected component 12 has 7 nodes
Connected component 13 has 5 nodes
Connected component 14 has 11 nodes
Connected component 15 has 8 nodes
Connected component 16 has 5 nodes
Connected component 17 has 6 nodes
Connected component 18 has 5 nodes
Connected component 19 has 5 nodes
Connected component 20 has 4 nodes
Connected component 21 has 5 nodes
Connected component 22 has 6 nodes

CLUSTERS
========
The graph has 174 clusters
Cluster 0 has 39 nodes
Cluster 1 has 12 nodes
Cluster 2 has 39 nodes
Cluster 3 has 40 nodes
Cluster 4 has 12 nodes
Cluster 5 has 40 nodes
Cluster 6 has 12 nodes
Cluster 7 has 38 nodes
Cluster 8 has 39 nodes
Cluster 9 has 40 nodes
Cluster 10 has 12 nodes
Cluster 11 has 38 nodes
Cluster 12 has 25 nodes
Cluster 13 has 38 nodes
Cluster 14 has 39 nodes
Cluster 15 has 40 nodes
Cluster 16 has 40 nodes
Cluster 17 has 39 nodes
Cluster 18 has 25 nodes
Cluster 19 has 25 nodes
Cluster 20 has 38 nodes
Cluster 21 has 12 nodes
Cluster 22 has 40 nodes
Cluster 23 has 40 nodes
Cluster 24 has 40 nodes
Cluster 25 has 25 nodes
Cluster 26 has 39 nodes
Cluster 27 has 39 nodes
Cluster 28 has 12 nodes
Cluster 29 has 37 nodes
Cluster 30 has 40 nodes
Cluster 31 has 25 nodes
Cluster 32 has 39 nodes
Cluster 33 has 25 nodes
Cluster 34 has 25 nodes
Cluster 35 has 38 nodes
Cluster 36 has 12 nodes
Cluster 37 has 12 nodes
Cluster 38 has 38 nodes
Cluster 39 has 40 nodes
Cluster 40 has 40 nodes
Cluster 41 has 39 nodes
Cluster 42 has 40 nodes
Cluster 43 has 39 nodes
Cluster 44 has 12 nodes
Cluster 45 has 38 nodes
Cluster 46 has 25 nodes
Cluster 47 has 25 nodes
Cluster 48 has 25 nodes
Cluster 49 has 25 nodes
Cluster 50 has 12 nodes
Cluster 51 has 25 nodes
Cluster 52 has 40 nodes
Cluster 53 has 25 nodes
Cluster 54 has 25 nodes
Cluster 55 has 25 nodes
Cluster 56 has 38 nodes
Cluster 57 has 40 nodes
Cluster 58 has 39 nodes
Cluster 59 has 12 nodes
Cluster 60 has 39 nodes
Cluster 61 has 25 nodes
Cluster 62 has 39 nodes
Cluster 63 has 12 nodes
Cluster 64 has 39 nodes
Cluster 65 has 25 nodes
Cluster 66 has 40 nodes
Cluster 67 has 38 nodes
Cluster 68 has 12 nodes
Cluster 69 has 25 nodes
Cluster 70 has 39 nodes
Cluster 71 has 39 nodes
Cluster 72 has 40 nodes
Cluster 73 has 40 nodes
Cluster 74 has 25 nodes
Cluster 75 has 25 nodes
Cluster 76 has 40 nodes
Cluster 77 has 12 nodes
Cluster 78 has 25 nodes
Cluster 79 has 25 nodes
Cluster 80 has 39 nodes
Cluster 81 has 39 nodes
Cluster 82 has 25 nodes
Cluster 83 has 40 nodes
Cluster 84 has 25 nodes
Cluster 85 has 39 nodes
Cluster 86 has 25 nodes
Cluster 87 has 25 nodes
Cluster 88 has 39 nodes
Cluster 89 has 25 nodes
Cluster 90 has 25 nodes
Cluster 91 has 25 nodes
Cluster 92 has 39 nodes
Cluster 93 has 40 nodes
Cluster 94 has 25 nodes
Cluster 95 has 39 nodes
Cluster 96 has 39 nodes
Cluster 97 has 12 nodes
Cluster 98 has 25 nodes
Cluster 99 has 25 nodes
Cluster 100 has 25 nodes
Cluster 101 has 39 nodes
Cluster 102 has 39 nodes
Cluster 103 has 12 nodes
Cluster 104 has 25 nodes
Cluster 105 has 25 nodes
Cluster 106 has 25 nodes
Cluster 107 has 25 nodes
Cluster 108 has 38 nodes
Cluster 109 has 40 nodes
Cluster 110 has 40 nodes
Cluster 111 has 25 nodes
Cluster 112 has 39 nodes
Cluster 113 has 40 nodes
Cluster 114 has 40 nodes
Cluster 115 has 25 nodes
Cluster 116 has 12 nodes
Cluster 117 has 25 nodes
Cluster 118 has 39 nodes
Cluster 119 has 39 nodes
Cluster 120 has 40 nodes
Cluster 121 has 39 nodes
Cluster 122 has 25 nodes
Cluster 123 has 12 nodes
Cluster 124 has 40 nodes
Cluster 125 has 25 nodes
Cluster 126 has 40 nodes
Cluster 127 has 39 nodes
Cluster 128 has 40 nodes
Cluster 129 has 40 nodes
Cluster 130 has 39 nodes
Cluster 131 has 25 nodes
Cluster 132 has 25 nodes
Cluster 133 has 12 nodes
Cluster 134 has 38 nodes
Cluster 135 has 25 nodes
Cluster 136 has 39 nodes
Cluster 137 has 25 nodes
Cluster 138 has 12 nodes
Cluster 139 has 25 nodes
Cluster 140 has 25 nodes
Cluster 141 has 25 nodes
Cluster 142 has 25 nodes
Cluster 143 has 25 nodes
Cluster 144 has 12 nodes
Cluster 145 has 25 nodes
Cluster 146 has 25 nodes
Cluster 147 has 40 nodes
Cluster 148 has 39 nodes
Cluster 149 has 25 nodes
Cluster 150 has 12 nodes
Cluster 151 has 12 nodes
Cluster 152 has 25 nodes
Cluster 153 has 25 nodes
Cluster 154 has 25 nodes
Cluster 155 has 25 nodes
Cluster 156 has 40 nodes
Cluster 157 has 25 nodes
Cluster 158 has 39 nodes
Cluster 159 has 25 nodes
Cluster 160 has 39 nodes
Cluster 161 has 39 nodes
Cluster 162 has 25 nodes
Cluster 163 has 12 nodes
Cluster 164 has 25 nodes
Cluster 165 has 40 nodes
Cluster 166 has 25 nodes
Cluster 167 has 38 nodes
Cluster 168 has 12 nodes
Cluster 169 has 25 nodes
Cluster 170 has 12 nodes
Cluster 171 has 40 nodes
Cluster 172 has 40 nodes
Cluster 173 has 39 nodes

Visualizing the graph:
```



rdf-network-analysis

May 15, 2020

1 Network Analysis of RDF Graphs

In this notebook we provide basic facilities for performing network analyses of RDF graphs easily with Python rdflib and networkx

We do this in 4 steps: 1. Load an arbitrary RDF graph into rdflib 2. Get a subgraph of relevance (optional) 3. Convert the rdflib Graph into an networkx Graph, as shown here 4. Get an network analysis report by running networkx's algorithms on that data structure

1.1 0. Preparation

[13]:
```
# Install required packages in the current Jupyter kernel
# Uncomment the following lines if you need to install these libraries
# If you run into permission issues, try with the --user option
import sys
# !pip install -q rdflib networkx matplotlib scipy
!{sys.executable} -m pip install rdflib networkx matplotlib scipy --user

# Imports
from rdflib import Graph as RDFGraph
from rdflib.extras.external_graph_libs import rdflib_to_networkx_graph
import networkx as nx
from networkx import Graph as NXGraph
import matplotlib.pyplot as plt
import statistics
import collections
```

```
Requirement already satisfied: rdflib in /home/amp/.local/lib/python3.8/site-
packages (5.0.0)
Requirement already satisfied: networkx in /home/amp/.local/lib/python3.8/site-
packages (2.4)
Requirement already satisfied: matplotlib in
/home/amp/.local/lib/python3.8/site-packages (3.2.1)
Requirement already satisfied: scipy in /home/amp/.local/lib/python3.8/site-
packages (1.4.1)
Requirement already satisfied: pyparsing in /usr/lib/python3/dist-packages (from
rdflib) (2.4.6)
Requirement already satisfied: six in /usr/lib/python3/dist-packages (from
rdflib) (1.14.0)
```

```
Requirement already satisfied: isodate in /home/amp/.local/lib/python3.8/site-
packages (from rdflib) (0.6.0)
Requirement already satisfied: decorator>=4.3.0 in /usr/lib/python3/dist-
packages (from networkx) (4.4.2)
Requirement already satisfied: kiwisolver>=1.0.1 in
/home/amp/.local/lib/python3.8/site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: numpy>=1.11 in /usr/lib/python3/dist-packages
(from matplotlib) (1.17.4)
Requirement already satisfied: cycler>=0.10 in
/home/amp/.local/lib/python3.8/site-packages (from matplotlib) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/lib/python3/dist-
packages (from matplotlib) (2.7.3)
```

1.2 1. Loading RDF

The first thing to do is to load the RDF graph we want to perform the network analysis on. By executing the next cell, we'll be asked to fill in the path to an RDF graph. This can be any path, local or online, that we can look up.

Any of the Turtle (`ttl.`) files that we include with this notebook will do; for example, `bsbm-sample.ttl`. But any Web location that leads to an RDF file (for example, the GitHub copy of that same file at https://raw.githubusercontent.com/albertmeronyo/rdf-network-analysis/master/bsbm-sample.ttl; or any other RDF file on the Web like https://raw.githubusercontent.com/albertmeronyo/lodapi/master/ghostbusters.ttl) will work too.

[18]:
```python
# RDF graph loading
path = input("Path or URI of the RDF graph to load: ")
rg = RDFGraph()
rg.parse(path, format='turtle')
print("rdflib Graph loaded successfully with {} triples".format(len(rg)))
```

```
Path or URI of the RDF graph to load: wechanged-german.ttl
rdflib Graph loaded successfully with 53 triples
```

1.3 2. Get a subgraph out of the loaded RDF graph (optional)

This cell can be skipped altogether without affecting the rest of the notebook; but it will be useful if instead of using the whole RDF grahp of the previous step, we just want to use a subgraph that's included in it.

By executing the next cell, we'll be asked two things:

- The URI of the ''entiy" type we are interested in (e.g. `http://dbpedia.org/ontology/Band`)
- The URI of the ''relation" connecting entities we are interested in (e.g. `http://dbpedia.org/ontology/influencedBy`)

Using these two, the notebook will replace the original graph with the subgraph that's constructed by those entity types and relations only.

```
[ ]: # Subgraph construction (optional)
     entity = input("Entity type to build nodes of the subgraph with: ")
     relation = input("Relation type to build edges of the subgraph with: ")

     # TODO: Use entity and relation as parameters of a CONSTRUCT query
     query = """
     PREFIX bsbm: <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/vocabulary/>
     CONSTRUCT {{ ?u a {} . ?u {} ?v }} WHERE {{ ?u a {} . ?u {} ?v }}""".
      ↪format(entity, relation, entity, relation)
     # print(query)
     subg = rg.query(query)

     rg = subg
```

1.4 3. Converting rdflib.Graph to networkx.Graph

Thanks to the great work done by the rdflib developers this step, which converts the basic graph data structure of rdflib into its equivalent in networkx, is straightforward. Just run the next cell to make our RDF dataset ready for network analysis!

```
[19]: # Conversion of rdflib.Graph to networkx.Graph
      G = rdflib_to_networkx_graph(rg)
      print("networkx Graph loaded successfully with length {}".format(len(G)))
```

```
networkx Graph loaded successfully with length 65
```

1.5 4. Network analysis

At this point we can run the network analysis on our RDF graph by using the networkx algorithms. Exeucting the next cell will output a full network analysis report, with the following parts:

- General network metrics (network size, pendants, density)
- Node centrality metrics (degree, eigenvector, betwenness). For these, averages, stdevs, maximum, minimum and distribution histograms are given
- Clustering metrics (connected components, clustering)
- Overall network plot

The report can be easily selected and copy-pasted for further use in other tools.

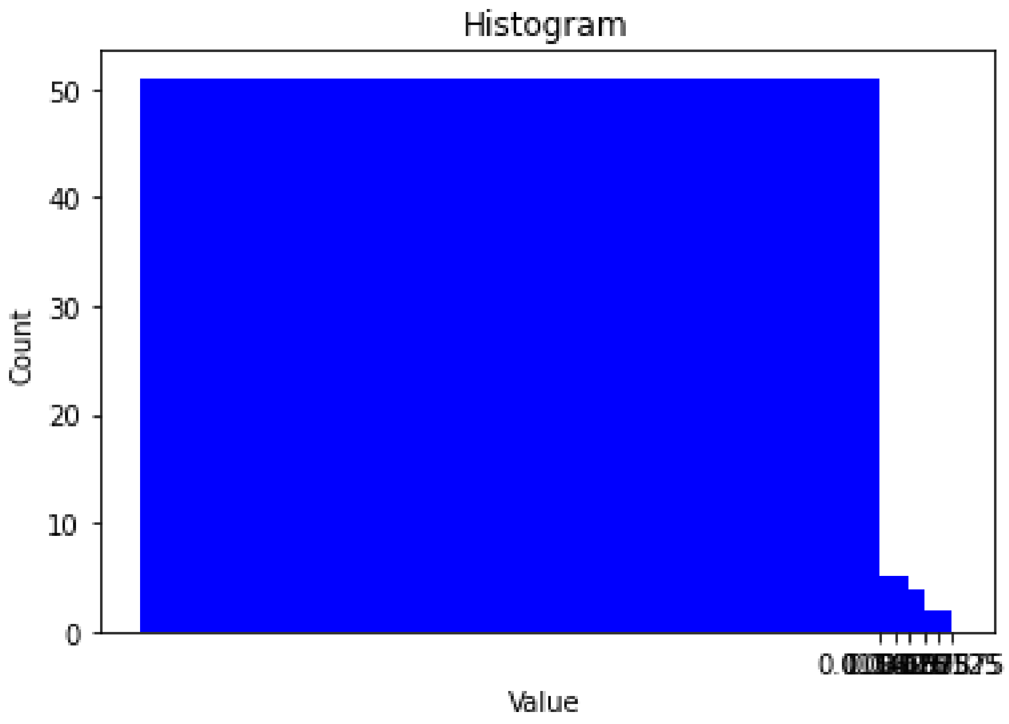
```
[20]: # Analysis

      def mean(numbers):
          return float(sum(numbers)) / max(len(numbers), 1)

      def number_of_pendants(g):
          """
          Equals the number of nodes with degree 1
          """
          pendants = 0
```

```
    for u in g:
        if g.degree[u] == 1:
            pendants += 1
    return pendants


def histogram(l):
    degree_sequence = sorted([d for n, d in list(l.items())], reverse=True)
    degreeCount = collections.Counter(degree_sequence)
    deg, cnt = zip(*degreeCount.items())
    print(deg, cnt)

    fig, ax = plt.subplots()
    plt.bar(deg, cnt, width=0.80, color='b')

    plt.title("Histogram")
    plt.ylabel("Count")
    plt.xlabel("Value")
    ax.set_xticks([d + 0.4 for d in deg])
    ax.set_xticklabels(deg)

    plt.show()

# Network size
print("NETWORK SIZE")
print("============")
print("The network has {} nodes and {} edges".format(G.number_of_nodes(), G.
 ↪number_of_edges()))
print()

# Network size
print("PENDANTS")
print("============")
print("The network has {} pendants".format(number_of_pendants(G)))
print()

# Density
print("DENSITY")
print("============")
print("The network density is {}".format(nx.density(G)))
print()

# Degree centrality -- mean and stdev
dc = nx.degree_centrality(G)
degrees = []
for k,v in dc.items():
    degrees.append(v)
```

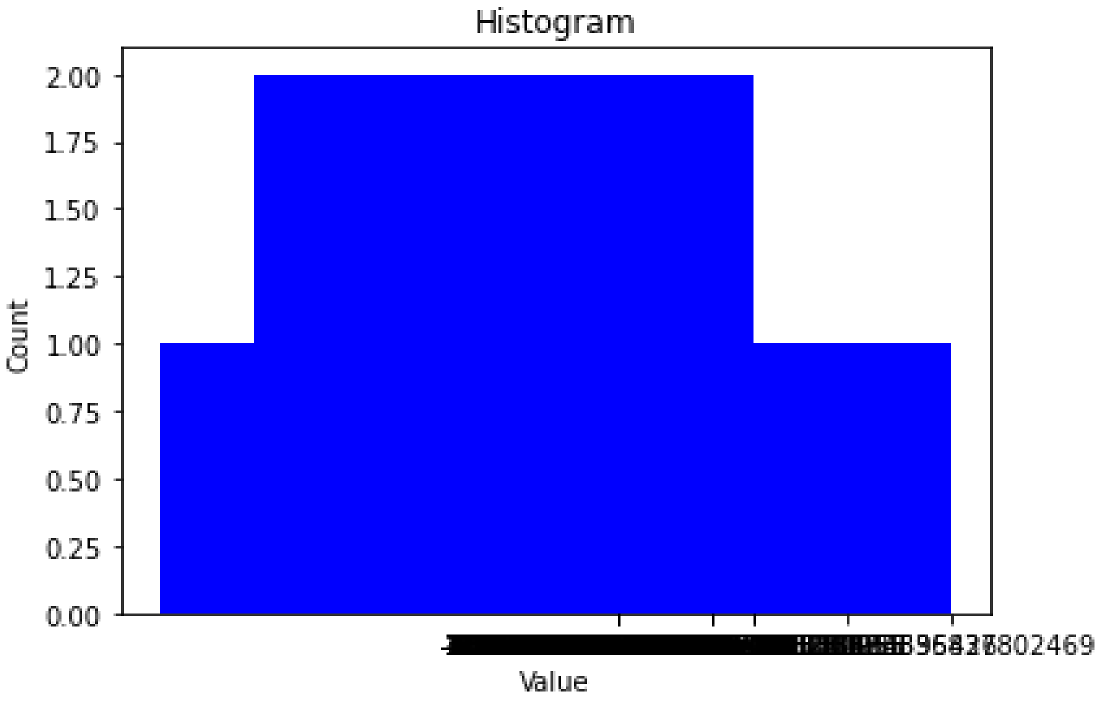
```
print("DEGREE CENTRALITY")
print("=================")
print("The mean degree centrality is {}, with stdev {}".format(mean(degrees),␣
 ↪statistics.stdev(degrees)))
print("The maximum node is {}, with value {}".format(max(dc, key=dc.get),␣
 ↪max(dc.values())))
print("The minimum node is {}, with value {}".format(min(dc, key=dc.get),␣
 ↪min(dc.values())))
histogram(dc)
print()

# Eigenvector centrality -- mean and stdev
ec = nx.eigenvector_centrality_numpy(G)
degrees = []
for k,v in ec.items():
    degrees.append(v)

print("EIGENVECTOR CENTRALITY")
print("======================")
print("The mean network eigenvector centrality is {}, with stdev {}".
 ↪format(mean(degrees), statistics.stdev(degrees)))
print("The maximum node is {}, with value {}".format(max(ec, key=ec.get),␣
 ↪max(ec.values())))
print("The minimum node is {}, with value {}".format(min(ec, key=ec.get),␣
 ↪min(ec.values())))
histogram(ec)
print()

# Betweenness centrality -- mean and stdev
# bc = nx.betweenness_centrality(G)
# degrees = []
# for k,v in bc.items():
#     degrees.append(v)
# print("BETWEENNESS CENTRALITY")
# print("======================")
# print("The mean betwenness centrality is {}, with stdev {}".
 ↪format(mean(degrees), statistics.stdev(degrees)))
# print("The maximum node is {}, with value {}".format(max(bc, key=bc.get),␣
 ↪max(bc.values())))
# print("The minimum node is {}, with value {}".format(min(bc, key=bc.get),␣
 ↪min(bc.values())))
# histogram(bc)
# print()
```

```python
# Connected components
cc = list(nx.connected_components(G))
print("CONNECTED COMPONENTS")
print("====================")
print("The graph has {} connected components".format(len(cc)))
for i,c in enumerate(cc):
    print("Connected component {} has {} nodes".format(i,len(c)))
print()

# Clusters
cl = nx.clustering(G)
print("CLUSTERS")
print("========")
print("The graph has {} clusters".format(len(cl)))
for i,c in enumerate(cl):
    print("Cluster {} has {} nodes".format(i,len(c)))
print()

# Plot
print("Visualizing the graph:")
plt.plot()
plt.figure(1)
nx.draw(G, with_labels=False, font_weight='normal', node_size=60, font_size=8)
plt.figure(1,figsize=(120,120))
plt.savefig('example.png', dpi=1000)
```

```
NETWORK SIZE
============
The network has 65 nodes and 53 edges

PENDANTS
============
The network has 51 pendants

DENSITY
============
The network density is 0.02548076923076923

DEGREE CENTRALITY
=================
The mean degree centrality is 0.02548076923076923, with stdev
0.020774719730186218
The maximum node is http://www.wikidata.org/entity/Q165824, with value 0.09375
The minimum node is wcd_00814_id, with value 0.015625
(0.09375, 0.078125, 0.0625, 0.046875, 0.03125, 0.015625) (2, 2, 4, 5, 1, 51)
```



```
EIGENVECTOR CENTRALITY
======================
The mean network eigenvector centrality is 0.052190328754421186, with stdev
0.11339581003839311
The maximum node is http://www.wikidata.org/entity/Q165824, with value
0.5823336837802469
The minimum node is http://www.wikidata.org/entity/Q2653682, with value
-4.221865487293616e-16
(0.5823336837802469, 0.40110781684595426, 0.23773673088280958,
0.23773673088280955, 0.23773673088280953, 0.2377367308828095,
0.16375158051905314, 0.1637515805190531, 0.16375158051905309,
0.16375158051905306, 0.16375158051905303, 4.0375682011403296e-16,
3.867620361637153e-16, 2.423140802377901e-16, 2.1493997183573874e-16,
2.0826656295842276e-16, 1.8925354328681477e-16, 1.860937486981313e-16,
1.7617115305582745e-16, 1.667799235809163e-16, 1.5837406027033936e-16,
1.3001507409184495e-16, 1.2555668835368535e-16, 1.1354908529513395e-16,
1.0195879145351594e-16, 9.659051261599858e-17, 8.249547678468506e-17,
7.150789936020287e-17, 6.477197106861316e-17, 6.34748456895395e-17,
6.255159781101699e-17, 5.748386506242491e-17, 4.5070823959909377e-17,
4.028075437461217e-17, 3.191177899331292e-17, 1.7134628208983114e-17,
1.504830421598233e-17, 1.5222146334878828e-18, -1.2422309969230075e-18,
-8.56840964600123e-18, -1.0566991786028656e-17, -1.3380020371347866e-17,
-1.82290962330364e-17, -2.5459354322188953e-17, -2.9381498680853597e-17,
-5.607146449104495e-17, -6.208476377865393e-17, -6.278772145308986e-17,
-1.0752076302444307e-16, -1.0828082789917711e-16, -1.7349787989201016e-16,
-1.765445338168193e-16, -1.879654759105251e-16, -1.9130239507871805e-16,
-1.93439031608548e-16, -1.9861678449786301e-16, -1.9935548922841931e-16,
-2.3141199604139336e-16, -2.477318548940688e-16, -2.722433427921724e-16,
-3.7706856978891896e-16, -4.221865487293616e-16) (1, 1, 1, 1, 2, 2, 1, 1, 1, 2,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
```



```
CONNECTED COMPONENTS
====================
The graph has 12 connected components
Connected component 0 has 5 nodes
Connected component 1 has 7 nodes
Connected component 2 has 5 nodes
Connected component 3 has 4 nodes
Connected component 4 has 9 nodes
Connected component 5 has 4 nodes
Connected component 6 has 6 nodes
Connected component 7 has 6 nodes
Connected component 8 has 4 nodes
Connected component 9 has 7 nodes
Connected component 10 has 4 nodes
Connected component 11 has 4 nodes
```

```
CLUSTERS
========
The graph has 65 clusters
Cluster 0 has 37 nodes
Cluster 1 has 12 nodes
Cluster 2 has 38 nodes
Cluster 3 has 37 nodes
Cluster 4 has 37 nodes
Cluster 5 has 25 nodes
Cluster 6 has 37 nodes
Cluster 7 has 25 nodes
Cluster 8 has 39 nodes
Cluster 9 has 40 nodes
Cluster 10 has 39 nodes
Cluster 11 has 25 nodes
Cluster 12 has 25 nodes
Cluster 13 has 37 nodes
Cluster 14 has 25 nodes
Cluster 15 has 25 nodes
Cluster 16 has 39 nodes
Cluster 17 has 25 nodes
Cluster 18 has 25 nodes
Cluster 19 has 39 nodes
Cluster 20 has 12 nodes
Cluster 21 has 37 nodes
Cluster 22 has 12 nodes
Cluster 23 has 12 nodes
Cluster 24 has 25 nodes
Cluster 25 has 37 nodes
Cluster 26 has 12 nodes
Cluster 27 has 25 nodes
Cluster 28 has 25 nodes
Cluster 29 has 37 nodes
Cluster 30 has 12 nodes
Cluster 31 has 25 nodes
Cluster 32 has 39 nodes
Cluster 33 has 12 nodes
Cluster 34 has 25 nodes
Cluster 35 has 40 nodes
Cluster 36 has 25 nodes
Cluster 37 has 12 nodes
Cluster 38 has 40 nodes
Cluster 39 has 12 nodes
Cluster 40 has 25 nodes
Cluster 41 has 39 nodes
Cluster 42 has 25 nodes
Cluster 43 has 38 nodes
Cluster 44 has 12 nodes
Cluster 45 has 25 nodes
Cluster 46 has 25 nodes
Cluster 47 has 25 nodes
Cluster 48 has 25 nodes
Cluster 49 has 25 nodes
Cluster 50 has 25 nodes
Cluster 51 has 37 nodes
Cluster 52 has 25 nodes
Cluster 53 has 12 nodes
Cluster 54 has 25 nodes
Cluster 55 has 40 nodes
Cluster 56 has 40 nodes
Cluster 57 has 40 nodes
Cluster 58 has 25 nodes
Cluster 59 has 12 nodes
Cluster 60 has 38 nodes
Cluster 61 has 25 nodes
Cluster 62 has 12 nodes
Cluster 63 has 25 nodes
Cluster 64 has 25 nodes

Visualizing the graph:
```