Information extraction from the Shakespeare Drama Corpus

Silvie Cinková

2022-06-05

Contents

Chapter 1

Outline and main concepts

1.1 The introductory sermon

This course is organized by two departments at the Charles University in Prague: The Institute of the Czech National Corpus at the Faculty of Arts and the Institute of Formal and Applied Linguistics at the Faculty of Mathematics and Physics. These two departments have been tightly collaborating ever since, including constant personal fluctuation in both directions. We are partners in the H2020 project Computational Literary Studies Infrastructure, and we are on the mission to make you, domain experts in literary studies, fit for using the current Natural Language Processing (NLP) techniques to boost your scholarly research.

In this course you should learn to:

1. Set up your own corpus of texts with or without TEI-XML markup.
2. Edit and annotate your corpus.
3. Extract information from running texts - operationalize pragmatic concepts in language structures.
4. Implement your operationalizations in corpus searches: sequential & tree queries.
5. Interpret your search results with elementary statistical methods.

In all of these areas you are going to get the big picture with an overview of the current practice, as well as a hands-on experience.

We have built this course around Shakespeare's dramas as an example of digital editions of classic literary texts with a very complex structure. We are by no means literary scholars ourselves, let alone experts in Shakespeare. I beg your apology that we will be just emulating literary research and present admittedly

naive research questions. The advantage of our naive inquiries is that we will be able to keep our example cases simple, so that you will leave this course having mastered a minimum to build on when you unleash your intellectual creativity on your real cases.

Although this is explicitly a programming-free course, we are going to drag you through a mire of different code languages. They are not real programming languages, but you are going to find them disconcertingly similar to such. We are going to stretch your perseverance and frustration tolerance. But we promise you that after these three days, you will be able to make advanced textual searches, and it is entirely up to you whether you stick to interactive user interfaces or switch to using their Application Programming Interfaces (APIs) from your own scripts.

You are definitely going to experience the numerous limits of the currently available interactive tools. Maybe this is going to lead you to the conclusion that you'd better invest your time in acquiring the coding skills necessary to become independent of such tools - because you have already mastered something very similar to them anyway. This would be, after all, the best possible outcome of this course, and it is our secret hope!

Nowadays, it does not take a full-fledged computer scientist any more to extract things from a text collection, count occurrences, and visualize simple descriptive statistics in plots. It is no rocket science - more to say, it is the daily routine of millions of marketing guys or journalists. There are quite a few software libraries dedicated to NLP and text-mining in the two most common programming languages **Python** and **R**, and you do not need much more from the general programming. Learning to use these is going to give you wings!

1.2 Operationalizing your research questions

When you start to operationalize your research questions, it is inevitably going to lead you towards structured and outcome-oriented thinking. Instead of free floating associations that make you write brilliant essays, you are going to conceive your research questions as experiments or series of experiments, and you are going to develop a sense of quantitative expectations on your outcomes.

What are you doing when you are trying to operationalize a concept, based on data? Imagine you have a corpus of several hundreds of drama pieces with XML-TEI markup. The markup tells you about each scene which characters are present and about each line who the speaker is. You may have quite an abstract initial research question, for instance: *What makes us recognize a drama character as the only or one of few protagonists, that is, the character(s) crucial for the piece?* You come up with a hypothesis that it has to do with how active the character is in the piece. So the activity is the concept you have to extract from the plays and quantify its occurrence. Most likely, you come to the conclusion that you can model activity with how much the character speaks

and/or how much he appears on the stage. You need to have a corpus of plays of which you know the protagonists and want to see how they differ from the ordinary characters in each piece and whether the difference goes in the same direction for all pieces in your collection.

You may want to make a difference between a character just appearing and a character actually saying something in a given scene, and you need to extract both these more specific concepts from the data. When you determine the concrete text elements you are going to extract from the data, you **operationalize** your concepts. For the speaking activity, you determine that you are going to count the number of scenes a character appears in, the number of lines (speech acts) he speaks, and the number of words he utters. For the appearance activity, you count in how many scenes it occurred with all other characters individually. As a next step, you **implement** your operationalization, considering practical details, such as whether or not you would count punctuation marks as words. Only then you have experimental results to analyze and interpret, e.g. by exploratory statistical calculations and plot visualizations, which you comment accordingly. This concrete case has been adopted from Fischer et al. [2018].

When doing research based on extractions from data, you want to have full access to the data and the liberty to slice it and perform calculations over it according to your own ideas. No interactive tool truly gives you this liberty. It is absolutely necessary that you learn programming basics for this type of research. Interactive tools are like Lego's thematic brick sets: the expensive Taj Mahal set only builds a Taj Mahal miniature model according to a Lego designer's taste, whereas your your old bucket of generic bricks gives you the fun of building anything exactly the way you want it. One does not need to be orthodox in either way, though: you are in the best position to pursue original research with standard methods when you use a few thematic sets together with a bucketful of generic bricks to build a toolchain tailored to your data and your ideas.

Soon you are going to spend quite some time with the operationalization of cognitive concepts with linguistic structures. We will present you a few ideas along with the technical thematic brick sets we have prepared for you, one by one.

Chapter 2

What is in the language? Research ideas for the Shakespeare Drama Corpus

The inspiration for this summer school has been the DraCor project (Fischer et al. [2019]), and in particular its **Shakespeare Drama Corpus**. For the purposes of our course, let us consider the Shakespeare Drama Corpus from two different perspectives - and allow them to overlap:

1. Differences between modern English and early modern English (Shakespeare's time);
2. Operationalized concepts.

We will draw inspiration from David Crystal's book *Think on My Words. Exploring Shakespeare's Language* (Crystal [2008]) and its companion web site shakespeareswords.com. Another source of inspiration for our exercises is the linguistic work of Douglas Biber (Biber [2004],Biber and Conrad [2009a]).

The research ideas listed here refer to tools and query languages of which you are soon going to hear more. You are going to learn to use linguistic markup and query languages to properly implement your operationalizations, but even now you can informally consider what you would ask a corpus for. Hints: *words starting with…*, *X preceding Y, Y with an adjective X as attribute…*

2.1 Differences in language use

What we would associate as first with older texts would be "difficult words" and archaic orthography. That is correct, and you can find glossaries of such

words on shakespeareswords.com. But apart from this, Crystal lists a number of interesting differences in grammar.

2.1.1 The case of *beget*

Shakespeare is said to have coined many new words. Crystal exposes this as a myth; but he stresses Shakespeare's creativity in the word formation. In other words, Shakepeare has not invented so many entirely new words but rather stretched the derivation options of already existing words. Crystal has compared Shakespeare's alleged coinages to other documents from the same period and found earlier uses of many. Nevertheless, let us explore a "difficult" word and consider its behavior regarding word formation and collocations. Word formation and collocability are two important indices of its use and meaning.

One current dictionary definition of the verb *beget* says:

1. *to make something start to exist or to happen*
2. *an old word meaning 'to become the father of a child'.*

Questions:

1. Which morphological forms does this verb occur in?
2. Does it occur with any prefixes?
3. *Who* begets *what*? Try to think of all possible grammatical structures from where you could extract this information.
4. Make a qualitative comparison (that is, by eyeballing) of the usage of this word and its derivatives in Shakespeare and in the EEBO (Early English Books Online) corpus in the Kontext corpus manager.

2.1.2 The case of the negation prefix *un-*

Crystal has noticed that Shakespeare tended to use this prefix more boldly towards the end of his writing career, around 1600. With which parts of speech did Shakespeare use it most? Compare the result with the *Early English Books Online (EEBO)* and/or a modern English corpus in Kontext.

2.1.3 Double genitives

Sometimes Shakespeare expresses the genitive with both the preposition *of* and the ending *'s* (Saxon genitive)*: the young Gentleman* ***of*** *the Count Orsino****'s*** *is return'd .* How would you detect this phenomenon?

2.1.4 Absolute genitives

Shakespeare also likes to use the Saxon genitive without the possessed noun following: *For halfe thy wealth, it is Anthonio's.* How would you detect the

absolute genitive?

2.1.5 Double adjective grading

You can express the intensity of a property by **grading** the adjective it is denoted by. The increased intensity is rendered by either periphrastic (*more intelligent*) or inflectional constructions (*smart-smarter-smartest, good-better-best*), depending mainly on the length of the adjective. At Shakespeare's times, this system was not established yet, so you can see double grading like this:

most unkindest cut of all

for the more better assurance

2.1.6 Verb in singular despite plural subject

English 3rd person verb forms usually keep grammatical concord with their subject. Whenever the subject is in plural or in a coordination, it requires the plural verb form. Exceptions (collective nouns: *the police*) and transparent heads (*a lot of X*) even tend to invoke the plural verb form. Nevertheless, Shakespeare's texts contain many cases where third person singular occurs with semantically and even grammatically plural subjects. How would you detect these cases? Are there any regular patterns?

Examples:

my old bones akes

what cares these roarers for the name of King

the Duke is comming from the Temple, and there is two or three Lords & Ladies more married

Our Master and Mistresse seekes you

all disquiet, horror, and perturbation followes her.

2.1.7 Marked word order

English is an S-V-O language: stylistically unmarked statement sentences contain the subject, the predicate verb, and the possible object in this order. Also, copula sentences (e.g. *The meal is good*) start with the subject, continue with the copula predicate verb and end with the predicate noun/adjective. Shakespeare often breaks this word order. Try and find as many examples of stylistically marked word order as possible. Qualitative considerations: does Shakespeare use marked word order to achieve some stylistic effects, or rather just to conform to the rhythm?

2.2 Pragmatic Concepts

Information extraction from non-fiction texts, such as technical reports, scientific papers or legal documents, is usually about facts. In fiction and conversation, on the other hand, we usually concentrate on its pragmatic aspects: *how* is the author/speaker conveying their message? What means do they use to achieve a certain effect? Text pragmatics spans lexicon and grammar alike.

An important name in this context is Douglas Biber (Biber [1988a]; Biber and Conrad [2009b]), who has investigated the linguistic variation in texts by extracting 67 English linguistic patterns ("features") from 481 texts of different written and spoken genres. He found out that the features form co-occurrence clusters. For instance, when a speaker uses passive verb forms, he is likely to use nominalizations as well, whereas he is unlikely to use second person in the same communication. On the other hand, when he uses a lot of the first and second person, he is likely to use contracted verb forms along with them.

To identify these feature clusters, Biber applied a statistical method (**Multidimensional Analysis (MDA)**), and he interpreted them as **text dimensions**. The features actually form a multidimensional space, and when we extract and count these features for a text, we can place it somewhere into this multidimensional space. Functionally similar texts then tend to be located together in clusters.

Biber's approach to genre-constituting characteristics of texts is unique in that it is data-driven. It proceeds bottom-up, with no *a priori* genre definitions.

For our course, the most relevant component of his research is how Biber **operationalized** the concepts in his book *Variation across Speech and Writing* (Biber [1988b]), Appendix II, pages 211-245, which you have in your study materials.

Drawing on Biber's linguistic features, we can play around with a few features of which we can assume that they have a distinct communication function, and try to classify Shakespeare's plays or different characters (speakers) according to them. Strictly speaking, we hazard those *a priori* guesses about forms and functions exactly in the opposite way than Biber did, but let us do it for the sake of exercise and then qualitatively estimate whether these assumptions make sense in this corpus.

2.2.1 Narrativity

Narration is associated with past tense(s) and third person, also with copula predicates in the past tense, present progressive tense, temporal adverbs.

2.2.2 Descriptivity

Descriptivity is associated with adjectives and participles in attributive positions, verbs in the present tense, and copula predicates in the present tense.

2.2.3 Interactivity

Interactivity is associated with the second person, questions, vocatives, and imperatives.

2.2.4 Uncertainty

Uncertainty can be associated with questions, hedge expressions and indefinite pronouns (*maybe, basically, a bit, some*), some modal verbs (*may*) or conditional markers (*if, when, whether, would*)

2.2.5 Emotionality

Extremely short lines - Shakespeare wrote his verses in the iambic pentameter – that means, each regular verse is ten syllables long, with an unstressed syllable shifting with a stressed syllable five times. Lines (utterances by one speaker) often stretch over several verses, but get shorter in very dramatic or emotional situations. Then one verse is filled with several speakers' lines.

Interjections (look them up at shakespeareswords.com).

2.2.6 Expression of stance

There are three major grammatical devices used for the lexico-grammatical expression of stance in English: complement clause constructions, stance adverbials, and modal verbs (Biber et al. [2018]). Try and extract sentences like these:

For so I know he is, they know he is – a most arch heretic, a pestilence

I mean that with my soul I love thy daughter

I could find in my heart that I had not a hard heart

I learn in this letter that Don Pedro of Aragon comes this night to Messina.

or these:

It is a problem that you don't approve of this.

Chapter 3

Programmable Corpora and DraCor

It is time you get familiar with the data sets and the tools. Let us start with the data in the more technical sense.

The DraCor project (Fischer et al. [2019]) consists of a GitHub repository with TEI-encoded dramas in several languages, and an interface where you can pick a piece and extract predefined parts of it, such as the cast, stage directions, and all lines spoken by a given character. Besides, you can get relation data in several formats to build network graphs of the characters, where you can choose between several metrics to represent edges between them. The characters are even interlinked with their WikiData entries. The interface is primarily an API - Application Programming Interface. This means that it is meant to be accessed by other programs rather than directly by human users, but it has a nice built-in demo that shows you what the API transfers into your computer when your script asks for it (Figures 3.1 and 3.2).

This API has even been wrapped for direct use as a sets of functions for Python and R, the two most popular programming languages in the Digital Humanities. This is, for instance how little you would have to code to ask DraCor API to give you all lines by individual speakers in R:

```
romeojuliet <- get_play_spoken_text_bych(corpus = "shake", play="romeo-and-juliet")
```

In Python, this would not be a lot of coding either. If you do not program at all, you may ask what difference it makes when you do have to write code anyway. A huge one! Imagine you want to get all lines by Juliet from Shakespeare's *Romeo and Juliet*. DraCor's API wrapper gives you a function that just wants you to fill in two things: the name of the corpus and the name of the piece. By itself it enters the DraCor corpora on GitHub, inspects the TEI tags and extracts all

passages by each speaker. Programming this from scratch would mean that you learn how to use generic API commands (that would be easy), and if there had not been an API at all, you would have to parse the XML-TEI yourselves with an XPATH query. XPATH is a query language to select elements and attributes in XML documents. If you already can write XPATH expressions, this would not be a big deal for you either, of course.

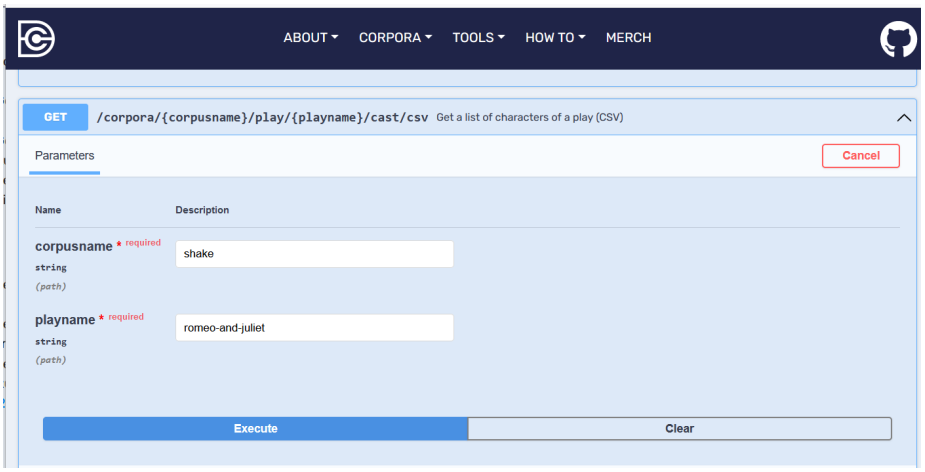

Figure 3.1: Ask for a comma-separated-values file with characters from Romeo and Juliet from the DraCor API online interface

You are going to fully appreciate DraCor's API once you get acquainted with the complex XML-TEI markup of the drama corpora!

Each piece in the DraCor corpora is one text (xml) file. It contains metadata, such as author, publication date, edition details, and languages occurring in the text, as well as metadata structuring the text itself, such as the frontmatter, cast list, stage instructions, scene and act indices, and speakers of individual lines. On top of that, it contains information about the gender of each character in the cast.

This is the Romeo item in the cast list:

```
<castItem sameAs="#Romeo_Rom">
            <role>
              <name>Romeo</name>
            </role>
          </castItem>
```

This is the word *love*:

```
<w xml:id="fs-rom-0001350" n="PRO.9" lemma="love" ana="#n1">love</w>
```

And this is a line by the Capulet servant Gregory in TEI-XML:

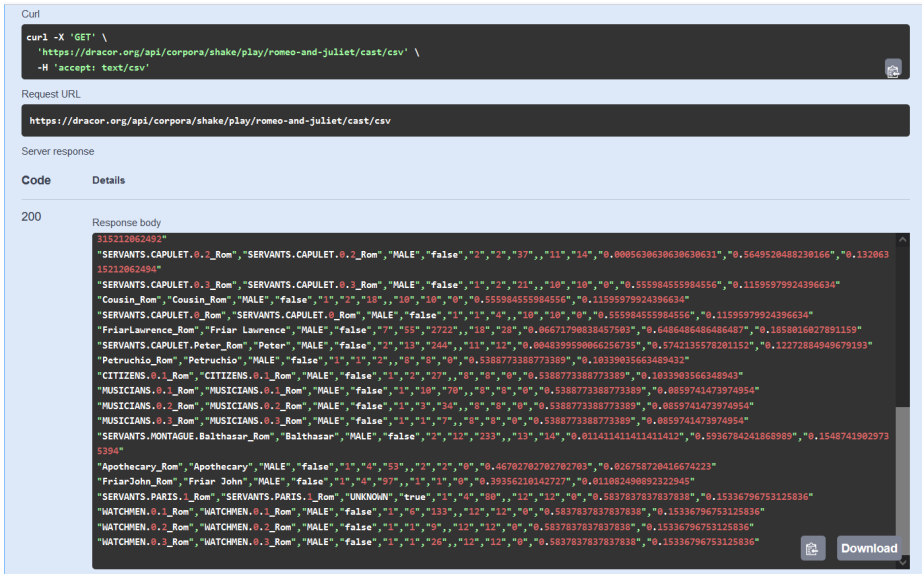

Figure 3.2: A comma-separated-values file with characters from Romeo and Juliet from the DraCor API online interface

```
    <sp xml:id="sp-0028" who="#SERVANTS.CAPULET.Gregory_Rom">
              <speaker xml:id="spk-0028">
                <w xml:id="fs-rom-0004720">GREGORY</w>
              </speaker>
              <p xml:id="p-0028">
                <lb xml:id="ftln-0028" n="1.1.14"/>
                <w xml:id="fs-rom-0004730" n="1.1.14" lemma="that" ana="#cs">That</w>
<c> </c>
                <w xml:id="fs-rom-0004750" n="1.1.14" lemma="show" ana="#vvz">shows</w>
<c> </c>
                <w xml:id="fs-rom-0004770" n="1.1.14" lemma="thou" ana="#pno">thee</w>
<c> </c>
                <w xml:id="fs-rom-0004790" n="1.1.14" lemma="a" ana="#d">a</w>
<c> </c>
                <w xml:id="fs-rom-0004810" n="1.1.14" lemma="weak" ana="#j">weak</w>
<c> </c>
                <w xml:id="fs-rom-0004830" n="1.1.14" lemma="slave" ana="#n1">slave</w>
                <pc xml:id="fs-rom-0004840" n="1.1.14">,</pc>
```

This markup identifies individual words (each word gets its unique ID). It also provides each word with its dictionary base form (**lemma**) and a morphological tag in an attribute called `ana` by TEI-XML.

TEI-XML encoded documents often contain tags indicating borders between

sentences, new paragraphs, headers, or line ends (this is typical of poetry).

Chapter 4

Shakespeare Drama Corpus

The data we are mainly going to work with is the Shakespeare Drama Corpus from the DraCor project.

Tab. 4.1 shows the metadata of the Shakespeare Drama Corpus in DraCor. We see that it contains 37 plays with 1,433 characters in total. The item `sp` (as *speaker*) stands for lines (utterances) uttered by the characters (31,066 lines in total). The corresponding TEI-XML tag is also called `sp`. Similarly, the stage instructions are tagged as `stage`, and there are 10,450 such passages.

Table 4.1: Metadata of the Shakespeare Drama Corpus

info	meta
description	Derived from the Folger Shakespeare Library. Enhancements documented in our README at G
uri	https://dracor.org/api/corpora/shake
title	Shakespeare Drama Corpus
name	shake
repository	https://github.com/dracor-org/shakedracor
licence	CC BY-NC 3.0
licenceUrl	https://creativecommons.org/licenses/by-nc/3.0/deed.en_US
plays	37
characters	1433
male	797
female	116
text	37
sp	31066
stage	10450
updated	2022-05-03 00:55:50
wordcount.text	908286
wordcount.sp	876744

info	meta
wordcount.stage	41230

Tab 4.2 lists the pieces in the Shakespeare Drama Corpus. Note especially the `name` column. Whenever you restrict your search to a single drama piece, you should use the values of the `name` column. The API gives you much more information on each play than just word counts, which we do not display here. Mostly it helps to draw a network graph of the characters, which we are not going to do in this course.

Table 4.2: Selected metadata of Shakespeare Drama Corpus

name	title	wordCountText
a-midsummer-night-s-dream	A Midsummer Night's Dream	17772
all-s-well-that-ends-well	All's Well That Ends Well	25066
antony-and-cleopatra	Antony and Cleopatra	27119
as-you-like-it	As You Like It	23721
coriolanus	Coriolanus	29948
cymbeline	Cymbeline	30141
hamlet	Hamlet	32539
henry-iv-part-i	Henry IV, Part I	26290
henry-iv-part-ii	Henry IV, Part II	28461
henry-v	Henry V	27982
henry-vi-part-1	Henry VI, Part 1	23423
henry-vi-part-2	Henry VI, Part 2	27902
henry-vi-part-3	Henry VI, Part 3	26849
henry-viii	Henry VIII	26525
julius-caesar	Julius Caesar	21164
king-john	King John	22399
king-lear	King Lear	28192
love-s-labor-s-lost	Love's Labor's Lost	23188
macbeth	Macbeth	18667
measure-for-measure	Measure for Measure	23712
much-ado-about-nothing	Much Ado About Nothing	23002
othello	Othello	28452
pericles	Pericles	20050
richard-ii	Richard II	23965
richard-iii	Richard III	31765
romeo-and-juliet	Romeo and Juliet	26470
the-comedy-of-errors	The Comedy of Errors	16723
the-merchant-of-venice	The Merchant of Venice	22743
the-merry-wives-of-windsor	The Merry Wives of Windsor	24490
the-taming-of-the-shrew	The Taming of the Shrew	23021

name	title	wordCountText
the-tempest	The Tempest	18143
the-winter-s-tale	The Winter's Tale	26927
timon-of-athens	Timon of Athens	20261
titus-andronicus	Titus Andronicus	22153
troilus-and-cressida	Troilus and Cressida	28502
twelfth-night	Twelfth Night	21714
two-gentlemen-of-verona	Two Gentlemen of Verona	18845

Perhaps the most notorious pieces by Shakespeare are *Romeo and Juliet* and *Hamlet* (Tab. 4.3 and 4.4)[1]. Let us have a closer look at their metadata as provided by the DraCor API. Again, we display only a selection of attributes, omitting the network information, and only the top ten characters ranked by the number of words they have uttered.

Table 4.3: Selected metadata of Romeo and Juliet

id	gender	numOfScenes	numOfSpeechActs	numOfWords
Romeo_Rom	MALE	14	163	4620
Juliet_Rom	FEMALE	11	118	4265
FriarLawrence_Rom	MALE	7	55	2722
Nurse_Rom	FEMALE	11	90	2205
Capulet_Rom	MALE	9	50	2136
Mercutio_Rom	MALE	4	62	2100
Benvolio_Rom	MALE	7	63	1160
LadyCapulet_Rom	FEMALE	10	45	874
PrinceEscalus_Rom	MALE	3	16	584
Paris_Rom	MALE	5	23	540

Table 4.4: Selected metadata of Hamlet

id	gender	numOfScenes	numOfSpeechActs	numOfWords
Hamlet_Ham	NA	13	358	11613
Claudius_Ham	MALE	11	102	4042
Polonius_Ham	MALE	8	87	2655
Horatio_Ham	MALE	9	110	2073
Laertes_Ham	MALE	6	62	1446
Ophelia_Ham	NA	5	58	1183
Gertrude_Ham	NA	10	69	1050
Gravedigger_Ham	MALE	1	33	736

[1]Yes, some protagonists of *Hamlet* seem to lack the gender information!

id	gender	numOfScenes	numOfSpeechActs	numOfWords
Rosencrantz_Ham	NA	7	48	708
Ghost_Ham	NA	2	14	680

With the DraCor infrastructure you can slice the data as you like. For instance, you can compare utterances by male characters with utterances by female characters, or compare the pieces chronologically. However, the current DraCor, although it has some linguistic markup, does not help you investigate the texts within the utterances. This is likely to be added in the future, but the current situation of DraCor comes us just handy: we will explore options to find and count interesting phenomena in the texts outside the DraCor infrastructure!

Chapter 5

Corpus managers and query languages

When you want to carry out true *distant reading* (Moretti [2007]) or *macroanalysis* (JOCKERS [2013]), you need serious coding. We recommend you e.g. *Text Analysis with R for Students of Literature* (Jockers [2014]) to start with.

For many standard methods of literary computing you do not need the complex TEI-XML markup at all, after you have sliced your data the way you want. For instance, when you want to compare the vocabulary most typical of male speakers vs. female speakers in Shakespeare's dramas, you need two plain text documents, each containing all lines by speakers of either gender. You compute an appropriate statistic for each word and document (e.g. *tf*idf*) and list the ones with the highest scores for either document[1]. Or you may want to model the topics in the corpus, using libraries that implement clever algorithms such as *Latent Semantic Analysis*, or *Latent Dirichlet Allocation*.

These are so-called *bag-of-words* methods. They work exclusively with word frequencies and do not accept any additional markup. They do not care about the contexts of the individual words. You make them more powerful when you pre-process the texts by lemmatization. This is especially important in languages with rich inflection[2].

However, when you want to examine words in contexts, you can use dedicated tools – corpus managers – with their query languages. In this course, you are going to learn about the **Corpus Query Language (CQL)** and Tree Query Lan-

[1] In R, you can use a single text-mining software library - *tidytext* (Silge and Robinson [2016]).

[2] The TEI-XML markup in the Shakespeare Drama Corpora contains lemmas, but the API currently does not seem to let you choose that you want the lemmas rather than words, so you would either have to download and parse the original XML-TEI, or run the plain texts through a lemmatizer of your own choice. More about this in Chapter @ref() .

guages for syntactically annotated corpora, exemplified by **Grew** (Guillaume [2021]). For the CQL, you will have two corpus managers to choose from: the Corpus Workbench integrated in TEITOK (see Section 6) and Kontext, the corpus manager used at the Institute of the Czech National Corpus, here with the preloaded Dracor Shakespeare Drama Corpus.

These corpus/tree query languages help you navigate in the linguistic markup. The linguistic markup describes each word (token) and each sentence, helping you to extract different text phenomena ranging from word forms (e.g. verbs of motion in the past tense) to complex structures (e.g. negation). The corpus manager usually wants the text input in the so-called vertical format, where each word and all its accompanying information is on a separate line. As a corpus user, you will hardly ever encounter the vertical format. The corpus manager will display your query matches in a much more user-friendly way.

The TEI-XML markup is often technically difficult to reconcile with the linguistic markup. This is also the reason why most modern linguistic analysis tools would not accept TEI-XML encoded text on input. You would first have to clear the original text of the TEI-XML tags, run the linguistic analyzers on it, plant the text back into the original TEI-XML markup and integrate the linguistic markup into it as new TEI-XML tags. The TEI-XML encoded texts can be so diverse that the solution must be tailored to each corpus individually.

Since the 1990s, there have been two parallel universes: the TEI-XML digital editions and corpus linguistics. These two communities use different text-search techniques, and their design decisions hardly take into account the needs of the other community. Therefore you have a problem finding a tool that would fit both.

Chapter 6

TEITOK

6.1 Examples of corpus building and querying workflows with TEITOK

6.1.1 Editing

In this course we are going to work with TEI-XML documents in the TEITOK tool. TEITOK is a web-based platform for viewing, creating, and editing corpora with rich textual markup as well as with linguistic annotation. On the one hand, it accommodates multiple document layers and maintains links between them – for instance orthographic normalization and proofreading (6.4), or even an audio/video file (6.1) or a facsimile scan (6.2). On the other hand, you can take a text layer and process it with an NLP tool. The linguistic annotation becomes just another document layer. Internally, TEITOK uses its own format (which is TEI-XML compliant anyway), but you can export all the layers with their TEI-XML markup to pure TEI-XML.

When building corpora from audio files or digitized facsimiles, you often want to preserve this raw material along with its text transcript. There are dedicated software tools to produce such transcripts.

You can edit metadata (6.3) in templates, or individual words in the text, or even correct the spans of bounding boxes in the image layer - this is how HTR and OCR tools capture the images of what they recognize as words. By the way, the "TOK" in "TEITOK" refers to the term **token**. In corpus linguistics, tokens are also called **running words**. A token is everything that gets recognized as a word by a **text tokenizer** (of course you can correct its output manually). Also punctuation marks are tokens. A HTR/OCR tool also recognizes tokens, but using mainly graphical criteria. Ideally, the HTR/OCR outcome would be identical with that of a text tokenizer, but that is virtually never the case, and not only due to errors: where a word is divided by a hyphen to continue on the

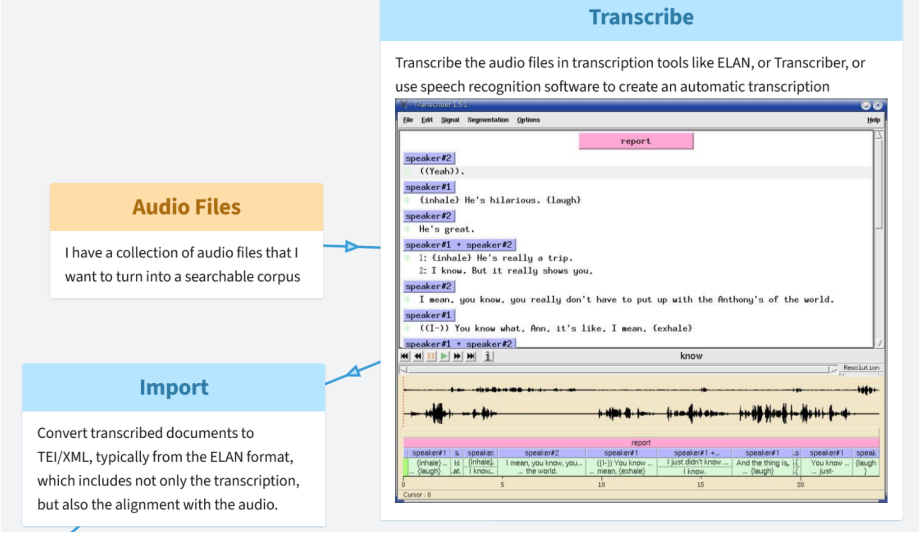

Figure 6.1: Audio files and transcripts in TEITOK

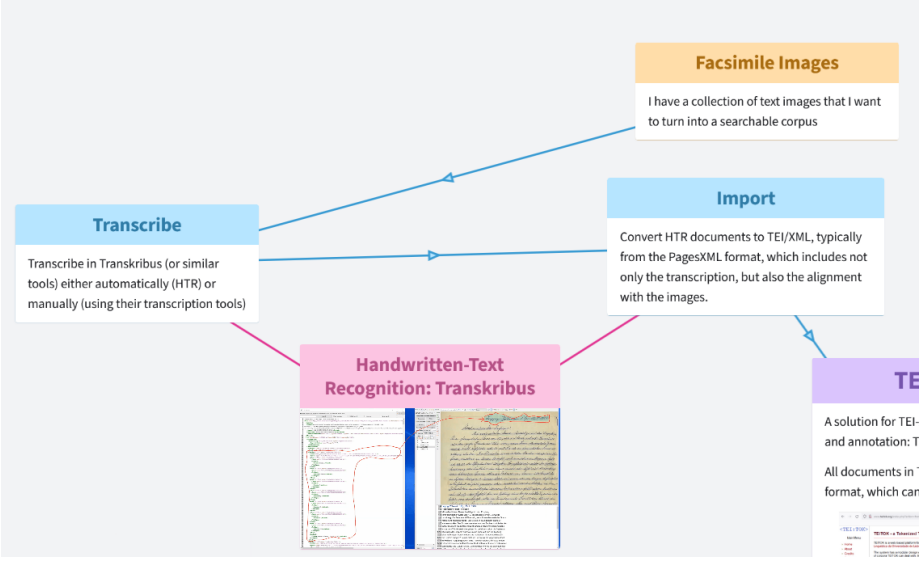

Figure 6.2: Handwritten Text Recognition transcripts in TEITOK

next line, the HTR/OCR tool correctly recognizes a token ending with a hyphen, the end of line, and another token on the start of the next line. Overcoming this inconsistency and being able to build further annotation on top of the linguistic tokens without losing the link to the graphical tokens is what makes TEITOK so useful.

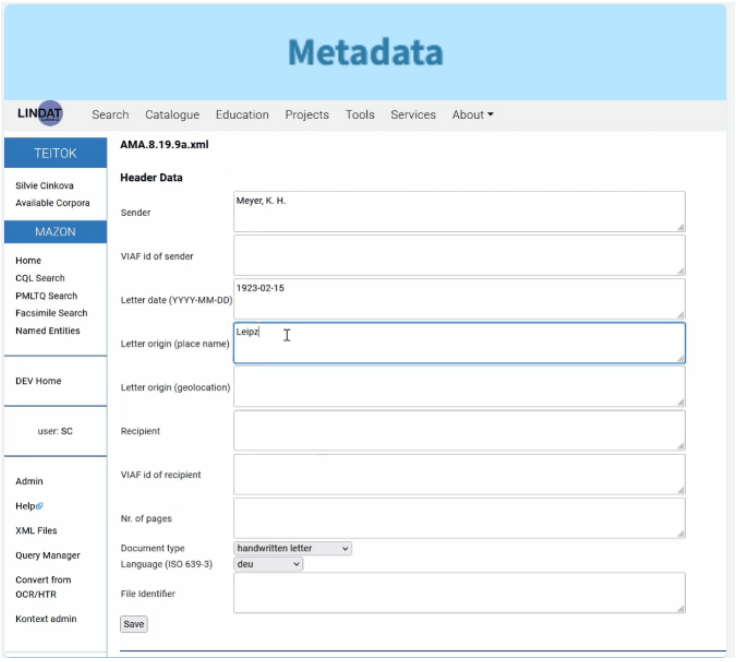

Figure 6.3: Edit different document layers in TEITOK

Each TEITOK token has an internal structure of attributes and values that you can edit. You can also manually merge, split, delete, and add tokens and create links from them to the corresponding segments (bounding boxes) in the source facsimile image 6.5, or to time indices in a source audio/video file.

HTR tools, e. g. Transkribus, sometimes give you a user-friendly option to train a dedicated model for your texts to be better recognized automatically. To train a model, you have to give the tool correctly recognized texts. You have to manually correct the automatic output or even make the transcript from scratch. The tools usually recognize individual word segments well, no matter whether or not they can interpret them in more detail. Normally you do not have to care about this part of recognition, but sometimes they fail doing even that, so you have to specifically correct this segmentation - draw correct rectangles around the individual segments (so-called bounding boxes). After you have corrected the bounding boxes of tokenization mishaps in TEITOK, you can export the data in exact the same format as the one produced by Transkribus. This means that you can use your further processed file for training an HTR.

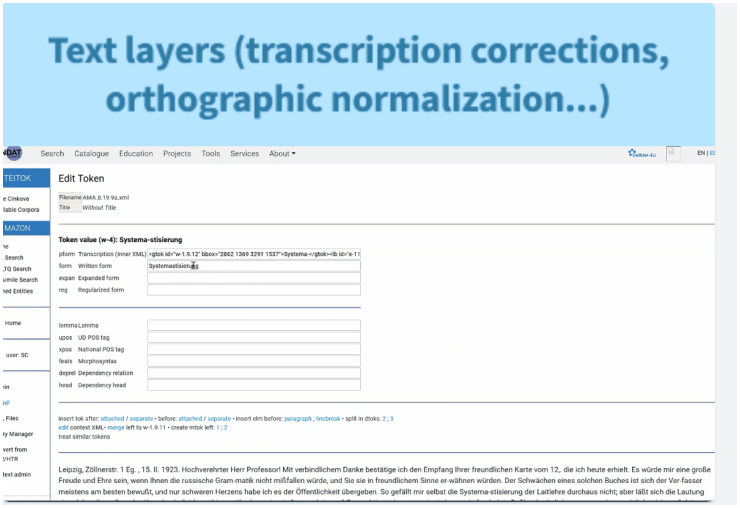

Figure 6.4: Edit different document layers in TEITOK

Figure 6.5: Edit bounding boxes in TEITOK

TEITOK can be linked to virtually any NLP tools. In most current projects it uses the **UDPipe** lemmatizer, morphological tagger and syntactic parser, and the **NameTag** Named-Entity Recognizer. You can run these tools directly in TEITOK and then manually check and correct their results if you wish to (6.6).

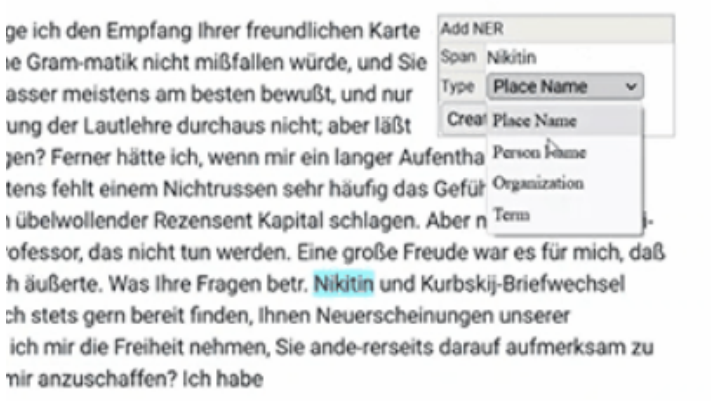

Figure 6.6: Manual annotation of named entities in TEITOK

When you have uploaded files already processed by a different tools with a different markup, you can create a look-up table in an ordinary spreadsheet editor for TEITOK to transform the original tags into **Universal Dependencies**, the annotation scheme used by UDPipe.

6.1.2 Querying and viewing

To query a corpus, TEITOK primarily uses the Corpus Workbench (CWB). It is designed to query large text corpora (up to 2 billion words). It was developed in the 1990s and has evolved to the corpus querying standard. Many current corpus managers have adopted the core CWB component, the Corpus Query Processor, and differ mainly in options of post-search calculation above the found matches. TEITOK offers CWB search along with a few post-search calculations. The most common sequential query language (without syntactic markup) is CWB's **CQL** (Corpus Query Language).

For syntactically annotated corpora, TEITOK currently integrates two tree query languages: **PMLTQ** and **Grew**.

During this course, you are going to learn how to query corpora both with CQL and a tree query language.

6.2 Using TEITOK in our course

Currently it is not yet easy enough to install and set up or plug in new tools. As an ordinary user without special programming skills, you have to ask the developer, Maarten Janssen, to set up and host a concrete TEITOK project for you. This is often an iterative process.

For this course, we have two dedicated TEITOK projects:

- **dracorshake** at https://quest.ms.mff.cuni.cz/teitok-dev/teitok/teaching/dracorshake/index.php
- **cls** at https://quest.ms.mff.cuni.cz/teitok-dev/teitok/teaching/cls/index.php

The dracorshake project contains the Shakespeare Drama Corpus and it is only meant for viewing and search.

The cls project, on the other hand, is empty, waiting for you to upload and experiment with your own documents.

The TEITOK administration does not work with user accounts but with individual projects. That means that you need to have set up a separate project for each corpus you want to build, and you obtain admin rights to it. You are going to learn to work with TEITOK in sessions dedicated to corpus building.

The query tools Corpus Workbench (for CQL search), Grew, and PMLTQ exist independently of TEITOK. It is just their instances that are plugged in. Learning to handle them gives you general skills beyond the TEITOK environment.

TEITOK is here to facilitate your corpus-building workflow and corpus search in the Shakespeare Drama Corpus. If you ever decide to build a corpus with TEITOK, it's setup is going to be tailored to your data and your research. New functions or new tools can be added ad hoc, and so can be the command links you see in the interface. You can as well try out a number of other tools. Therefore, do not think of the individual steps in your workflow in terms of interaction with TEITOK, but concentrate on what happens to the data in each step. Then you are going to be able to choose appropriate tools for your projects yourselves.

Chapter 7

NLP tools

7.1 Tokenizer, lemmatizer, tagger, parser

In digital editions, TEI-XML tags primarily indicate text parts based on their typography (*bold, heading, bullet list, page break*) and genre-related structural elements (*chapter, stage, cast, frontpage*). Moreover, TEI-XML allows to encode linguistic annotation. The most common linguistic markup determines individual tokens and sentence borders. Furthermore, each token can contain an entire series of TEI-XML attributes to store additional tags. Many NLP tools consist of this pipeline: tokenization, lemmatization, tagging, and parsing. This chapter is dedicated to this additional markup, not considering its implementation in TEI-XML.

When a text is lemmatized, it contains the basic dictionary form of each token. The exact outcome is specific to the linguistic tradition the given tool has adopted. Most differences occur in function words and derived words. For instance, personal pronouns can either have separate lemmas for each gender, or be all lemmatized as masculine; negated words (e.g. *unavailable*) can either be lemmatized with or without the negative suffix.

Morphological tagging carries information about each token's part of speech and some details relevant to the given part of speech. For instance tense in verbs. The details are again specific to a language and its linguistic traditions.

The crown of the linguistic markup is syntactic parsing. It determines relations between words within one sentence. You might have hated it at school, but it gives you a tremendous power when extracting information from the content, especially in languages with rich inflection and free word order. It is very often used to extract events along with their participants, circumstances and relations between entities.

This NLP workflow is so common that it is typically combined in a single tool.

7.2 Named-Entity Recognizer

Persons, Places, Times, Dates, Organizations, Countries, Currencies… these are Named Entities, and their importance for information extraction is obvious.

7.3 UDPipe and NameTag

The NLP tools we are going to use in this course are UDPipe and NameTag. Both are multilingual, state-of-the art tools based on neural networks. UDPipe is currently available for more than 70 languages. NameTag currently works well for Czech, Dutch, English, German, and Spanish, and a truly multilingual release is on its way. UDPipe works with a multilingual tagset called Universal Dependencies.

UDPipe's performance differs between languages, depending on the morphosyntactic complexity of its grammar related to the size of the training data available. You may find a better parser for your language than UDPipe, if that parser uses data not available to UDPipe or is combined with a morphological lexicon. This is currently the case of Hungarian, to name just one example.

In any case, if your parser is not outputting Universal Dependencies tags, you would benefit of mapping the outcome to the Universal Dependencies tagset, since Universal Dependencies have long become an established standard, and using them is going to make your corpus more interoperable.

7.4 What to do when the tools are performing poorly

All modern NLP tools are based on machine-learning. They have been trained on manually annotated texts to learn the correct human decisions in the linguistic markup. Typically they have been trained on contemporary press and Wikipedia, possibly some fiction. This is their domain, on which they also have been evaluated. So it can well happen that you are using a tool with a declared performance over 99%, but its performance on your data is much lower. This occurs when your data are very different from what the tool has been trained on. For instance, it can be systematically messing up the second person, because it has hardly encountered it during training. The same goes for archaic orthography, specific vocabulary, versed and rhymed language, and so on.

When using UDPipe, you can easily help it - and contribute to the entire community! You just take a sample of your data of about 5,000 tokens, process it with the NLP tool and manually correct the annotation. Then you upload it to the developers' repository and kindly ask them to include your data set into their training data. There is a standard workflow for that with UDPipe.

You will get acquainted with the linguistic annotation in a dedicated session,

as an exercise of your corpus querying skills. Doing a little bit of annotation is the most efficient way to get familiar with the linguistic patterns, and to memorize the tags. Within the CLS Infra Project, we love to award 3-month Transnational Access mentored fellowships in Prague for just such projects. If you are interested, check out the TNA Section on the CLS Infra website.

Bibliography

Douglas Biber. *Variation across Speech and Writing*. Cambridge University Press, Cambridge, 1988a. doi: 10.1017/CBO9780511621024. URL https://www.cambridge.org/core/books/variation-across-speech-and-writing/A546CF5ED8F8E62F1432CB2F369CF356. DOI: 10.1017/CBO9780511621024.

Douglas Biber. *Variation across Speech and Writing*. Cambridge University Press, Cambridge, 1988b. doi: 10.1017/CBO9780511621024. URL https://www.cambridge.org/core/books/variation-across-speech-and-writing/A546CF5ED8F8E62F1432CB2F369CF356. DOI: 10.1017/CBO9780511621024.

Douglas Biber. 7th international conference on the statistical analysis of textual data. Louvain, 2004. Presses universitaires de Louvain.

Douglas Biber and Susan Conrad. *Register, Genre, and Style*. Cambridge Textbooks in Linguistics. Cambridge University Press, Cambridge, 2009a. doi: 10.1017/CBO9780511814358. URL https://www.cambridge.org/core/books/register-genre-and-style/55217DB70BD87C93FA2BBA70DB4C2575. DOI: 10.1017/CBO9780511814358.

Douglas Biber and Susan Conrad. *Register, Genre, and Style*. Cambridge Textbooks in Linguistics. Cambridge University Press, Cambridge, 2009b. doi: 10.1017/CBO9780511814358. URL https://www.cambridge.org/core/books/register-genre-and-style/55217DB70BD87C93FA2BBA70DB4C2575. DOI: 10.1017/CBO9780511814358.

Douglas Biber, Jesse Egbert, and Meixiu Zhang. *Lexis and grammar as complementary discourse systems for expressing stance and evaluation*. Number 296 in Pragmatics & Beyond New Series. John Benjamins Publishing Company, 2018. URL https://doi.org/10.1075/pbns.296.08bib.

David Crystal. *Think On My Words: Exploring Shakespeare's Language*. Cambridge University Press, Cambridge, 2008. doi: 10.1017/CBO9780511755095. URL https://www.cambridge.org/core/books/think-on-my-words/03034E937E6D44283CD93E6576FC36BD. DOI: 10.1017/CBO9780511755095.

Frank Fischer, Peer Trilcke, Christopher Kittel, Carsten Milling, and Daniil Skorinkin. Digital Humanities 2018: Puentes/Bridges. Mexico: Red de Hu-

manidades Digitales A. C., 2018. Mexico: Red de Humanidades Digitales A. C.

Frank Fischer, Ingo Börner, Mathias Göbel, Angelika Hechtl, Christopher Kittel, Carsten Milling, and Peer Trilcke. Digital Humanities 2019. Utrecht, The Netherlands, 2019.

Bruno Guillaume. Eacl 2021 - 16th conference of the european chapter of the association for computational linguistics. 2021.

MATTHEW L. JOCKERS. *Macroanalysis: Digital Methods and Literary History*. University of Illinois Press, 2013. URL http://www.jstor.org/stable/10.5406/j.ctt2jcc3m.

Matthew L. Jockers. *Text Analysis with R for Students of Literature*. Springer, 2014.

F. Moretti. *Graphs, Maps, Trees: Abstract Models for Literary History*. Verso, 2007. URL https://books.google.cz/books?id=MDxoGQAACAAJ. LCCN: 2005017437.

Julia Silge and David Robinson. tidytext: Text mining and analysis using tidy data principles in r. *JOSS*, 1(3), 2016. doi: 10.21105/joss.00037. URL http://dx.doi.org/10.21105/joss.00037.