Lerneinheit: CATMA-Annotationen auswerten mit GitMA

Mareike Schumacher [1]
Michael Vauth
1. Universität Regensburg

Thema:	Manuelle Annotation	DOI:	10.48694/fortext.3753
Jahrgang:	1	Ausgabe:	4
Erscheinungsdatum:	2024-08-07	Erstveröffentlichung:	2022-03-07 auf fortext.net
Lizenz:			open access

Allgemeiner Hinweis: Rot dargestellte Begriffe werden im Glossar am Ende des Beitrags erläutert. Alle externen Links sind auch am Ende des Beitrags aufgeführt.

Eckdaten der Lerneinheit

- Anwendungsbezug: Franz Kafkas *Urteil*
- Methodik: Digitale, manuelle Annotation
- Angewendetes Tool: CATMA und GitHub sowie das Python-Package GitMA
- Lernziele: Laden eines Demo-Projektes aus CATMA, Überblick über Annotationen verschaffen, Annotationsdaten in Tabellen visualisieren, Goldstandard erstellen, IAA berechnen
- Dauer der Lerneinheit: 3 Stunden
- Schwierigkeitsgrad des Tools: mittel

1. Anwendungsbeispiel

Dieser Lerneinheit liegen CATMA-Annotationen (vgl. Annotation) zu Franz Kafkas *Urteil* zu Grunde. Die leitende Fragestellung der insgesamt zwei Annotatoren war: In welchen Textpassagen treten statische und prozesshafte Ereignisse auf? Zusätzlich zu den Annotationskategorien konnten die Eigenschaften „mental“, „persistent“, „intentional“ und „vorhersehbar“ jeweils mit einer binären ja-nein-Wertigkeit vergeben werden. Für die Auswertung von CATMA-Annotationen eignet sich das Tool GitMA (Vauth u. a. 2022), ein Python-Package mit dem Sie direkt auf das Backend von CATMA (Schumacher 2024a) zugreifen können.

2. Vorarbeiten

Diese Lerneinheit wird mithilfe von Jupyter Notebooks (Kluyver u. a. 2016) ausgeführt. Ein solches Notebook ermöglicht es Ihnen, von uns vorbereitete kurze Code-Einheiten (vgl. Programmiercode) per Klick aufzurufen. Installieren Sie bitte vorab Anaconda (Anaconda Software Distribution 2020). Haben Sie Anaconda installiert, so nutzen Sie dieses Programm, um Git (Chacon und Straub 2014) zu installieren. Starten Sie dafür Anaconda und wechseln Sie links im Menü zu „Environments“. Geben Sie dann oben rechts in die Suchleiste „Git“ ein und wählen Sie im Drop-Down-Menü oben in der Mitte „All“. Im Suchergebnis erscheint nun eine Zeile für „Git – Distributed version control system“ vor der links ein leerer Kasten steht.

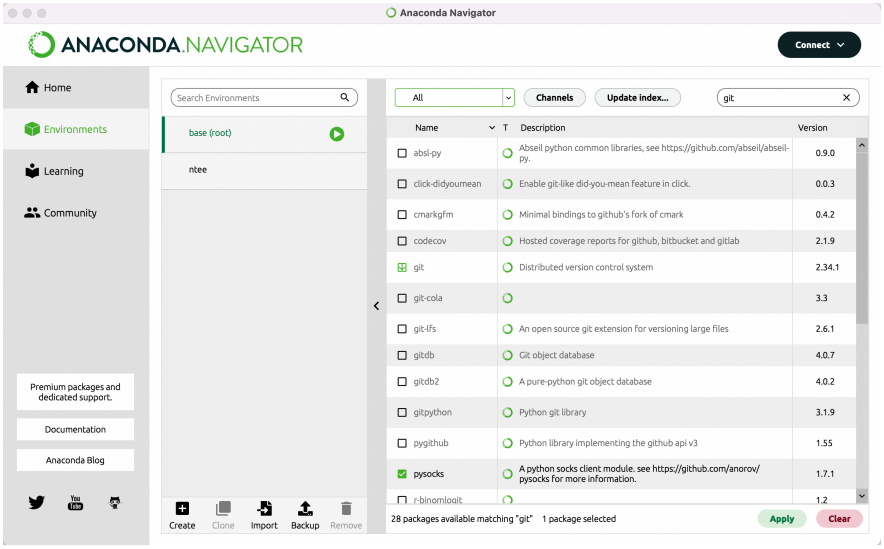

Abb. 1: Suche nach „Git“ innerhalb von Anaconda

Klicken Sie in diesen Kasten, sodass dort ein kleiner Pfeil erscheint (vgl. Abb. 1) und gehen Sie dann auf den Button „Apply“ unten rechts. Es kann einen Moment dauern bis das Programm Ihnen anzeigt, was die Installation von GitMA bedeutet, bzw. welche anderen Packages zusätzlich installiert oder aktualisiert werden müssen. Sobald Ihnen die Liste angezeigt wird, können Sie mit „Apply“ die Installation bestätigen (vgl. Abb. 2).

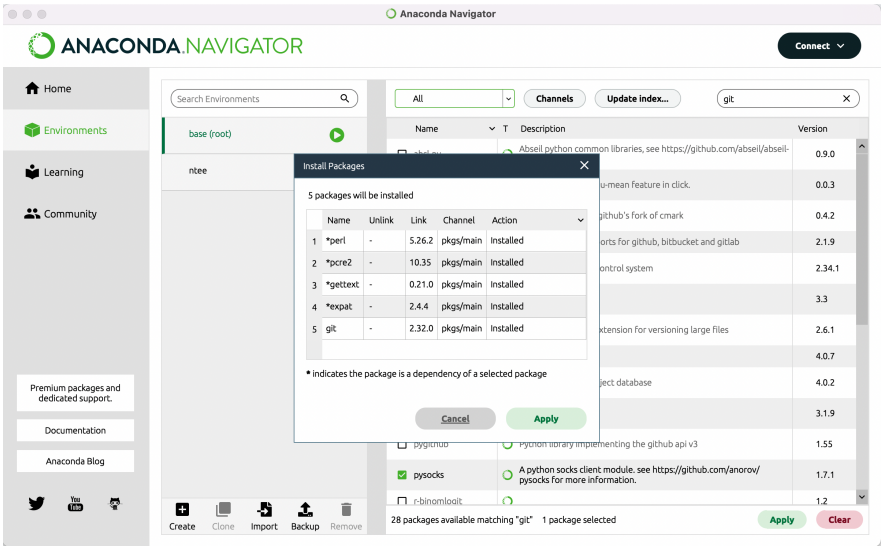

Abb. 2: Installation von Git innerhalb von Anaconda

Außer Git benötigen Sie für diese Lerneinheit noch das Python-Package „Pip“. Prüfen Sie, ob dieses Packet installiert werden muss, indem Sie oben rechts in der Suchleiste auf das Kreuz klicken. Geben Sie nun „Pip“ ein. Wenn im Suchergebnis vor „Pip – Pypa recommended tool for installing python packages“ noch kein grünes Häkchen steht, installieren Sie es ebenso wie Git. Im vierten Teil dieser Lerneinheit erfahren Sie, wie Sie bei einer kollaborativen Annotation (Jacke 2024a) die Übereinstimmung zwischen zwei oder mehr Annotator*innen in Form eines Gamma-Wertes bestimmen können. Ebenso wie GitMA kann das hierfür notwendige Python Package „Pygamma“ (Titeux und Riad 2021) nicht über die grafische Benutzeroberfläche installiert werden. Darum öffnen Sie nun bitte in Anaconda eine Commandline. Gehen Sie dazu links im Menü zurück zu „Environments“ und klicken Sie auf den Play-Button hinter „Base (root)“. Gehen Sie dann auf „Open Terminal“ (vgl. Abb 3).

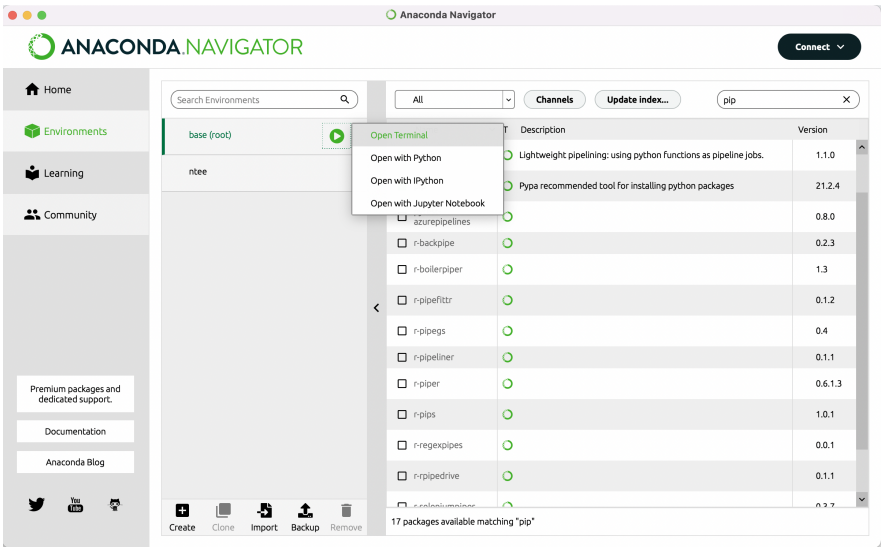

Abb. 3: Öffnen einer Commandline in Anaconda

Zuerst installieren Sie Pygamma, indem Sie folgenden Code kopieren, in die Kommandozeile einfügen und mit Enter bestätigen:

```
pip install pygamma-agreement
```

Um GitMA zu installieren, geben Sie dann bitte folgenden Code ein:

```
pip install git+<https://github.com/forTEXT/gitma>
```

Laden Sie nun unser CATMA-Demo-Projekt sowie die Jupyter Notebooks zur Nutzung von GitMA herunter. Gehen Sie dazu in unser GitMA-GitHub-Repository und dort zunächst auf den grünen Button „Code“ und dann auf „Download ZIP“ (vgl. Abb. 4). Wiederholen Sie das gleiche mit dem forTEXT.net-GitHub-Repository. Entpacken Sie dann die ZIP-Archive, indem Sie sie in Ihrem Ordnersystem ausfindig machen und dann doppelt draufklicken. Kopieren Sie aus dem forTEXT.net-Ordner die vier Notebooks „Annotationen auswerten mit GitMA 1-4“ in den Unterordner des GitMA-Ordners mit dem Namen „demo_notebooks“.

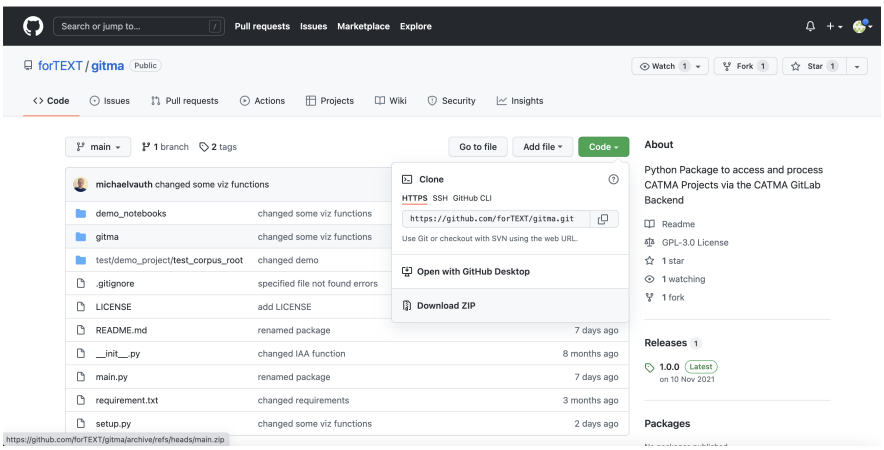

Abb. 4: Download des GitMA-Demo-Projektes und der dazugehörigen Jupyter Notebooks

Wenn Sie mit eigenen Daten arbeiten wollen, so müssen Sie diese zunächst in einer manuellen Annotation (Jacke 2024b) mit CATMA erstellen. Wie Sie solche Annotationsdaten anlegen können, erfahren Sie in unserer Lerneinheit zur manuellen Annotation mit CATMA (Horstmann 2024). Wenn Sie ausschließlich mit dem von uns bereitgestellten Demo-Projekt arbeiten wollen, so können Sie im nächsten Abschnitt direkt zur Funktion der Annotationsexploration springen. Wenn Sie erfahren möchten, wie Sie den Zugang zu Ihren eigenen CATMA-Projekten herstellen können, so fahren Sie einfach mit dem nächsten Schritt fort.

3. Funktionen

Mit GitMA können Sie zahlreiche Formen der Annotationsauswertung vornehmen. In dieser Lerneinheit zeigen wir Ihnen in vier thematischen Blöcken, wie Sie Projekte aus CATMA laden können, wie Sie Ihre Annotationsdaten explorieren, wie Sie die Erstellung eines Gold Standard unterstützen können und wie Sie ein Inter-Annotator-Agreement berechnen können.

Annotationsdaten aus CATMA laden

Wechseln Sie nun in Anaconda links im Menü von „Environments“ zu „Home“ und klicken Sie in der Kachel „Jupyter Notebook“ auf den „Launch“-Button. Ihnen wird nun in Ihrem Browser Ihr Ordnersystem angezeigt. Gehen Sie zu dem Ordner, in dem Sie das GitMA-Repository von GitHub gespeichert haben. Klicken Sie auf den Ordner „gitma-main“ und öffnen Sie dann den Ordner „Annotationen auswerten mit GitMA-Notebooks“, den Sie zuvor hierher verschoben haben (s.o.) und dann das erste Notebook „Annotationen auswerten mit GitMA 1“, indem Sie einfach darauf klicken. Das erste Notebook öffnet sich dann in einem neuen Browser-Tab (vgl. Abb. 5). Folgen Sie am besten den Anleitungen direkt im Notebook. Falls Sie eine Auswahl unserer Codebeispiele lieber zu einem eigenen Notenbook zusammenfassen wollen, finden Sie die einzelnen Schritte im Folgenden dokumentiert.

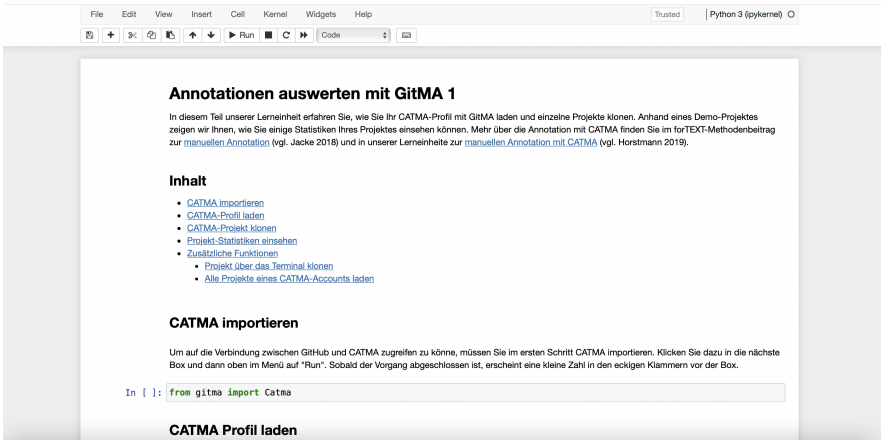

Abb. 5: Jupyter Notebook zum Laden von CATMA-Daten über GitMA

Im ersten Teil unserer Lerneinheit erfahren Sie, wie Sie Ihr CATMA-Profil mit GitMA laden und einzelne Projekte klonen. Außerdem zeigen wir Ihnen, wie Sie einige Statistiken Ihres Projektes einsehen können. Mehr über die Annotation mit CATMA finden Sie im forTEXT-Methodenbeitrag zur manuellen Annotation (Jacke 2024b) und in unserer Lerneinheit zur manuellen Annotation mit CATMA (Horstmann 2024).

1. CATMA importieren

Um auf die Verbindung zwischen GitHub und CATMA zugreifen zu können, müssen Sie im ersten Schritt CATMA importieren. Dazu können Sie folgenden Code nutzen:

```
from gitma import Catma
```

2. CATMA Profil laden

Sie können Ihr CATMA-Profil ganz einfach über ein sogenanntes Access Token für GitMA zugänglich machen. Um ein Acess Token zu erstellen, öffnen Sie Ihren CATMA-Account und klicken Sie auf das Profil-Icon oben rechts (vgl. Abb. 6). Es öffnet sich ein Menü, in dem Sie „Get Access Token“ auswählen können.

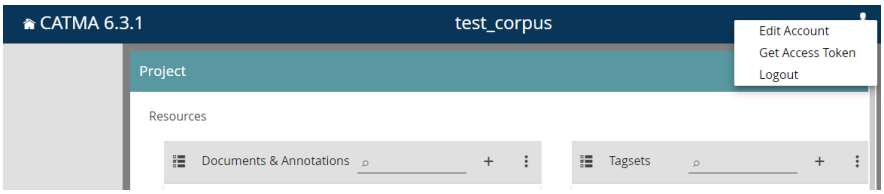

Abb. 6: Access Token im CATMA-Profil anfordern

Es öffnet sich nun ein Fenster, in dem Sie das Access Token benennen können. Sie können den Zugang Ihres Tokens zeitlich beschränken oder unbeschränkten Zugang gewähren (vgl. Abb. 7). Behalten Sie im Hinterkopf, dass mit diesem Token Zugang zu Ihren gesamten CATMA-Daten gewährt wird. Verwalten Sie in CATMA sensible Daten, so achten Sie darauf, das Access Token nicht weiterzugeben. Sie können immer wieder neue Access Tokens generieren.

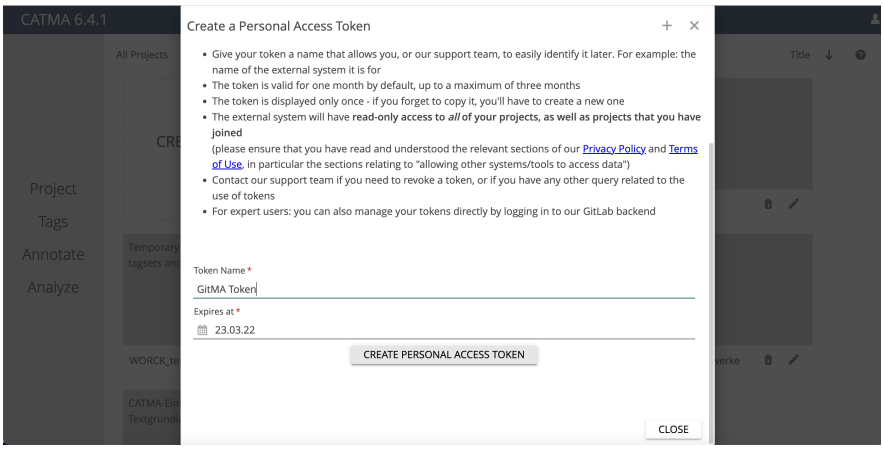

Abb. 7: Access Token benennen und zeitlich beschränken

Sie können Access Tokens auch im CATMA-GitLab generieren. Loggen Sie sich dazu auf CATMA GitLab mit Ihren CATMA-Account-Daten ein und nutzen Sie die API-Settings (vgl. Abb. 8).

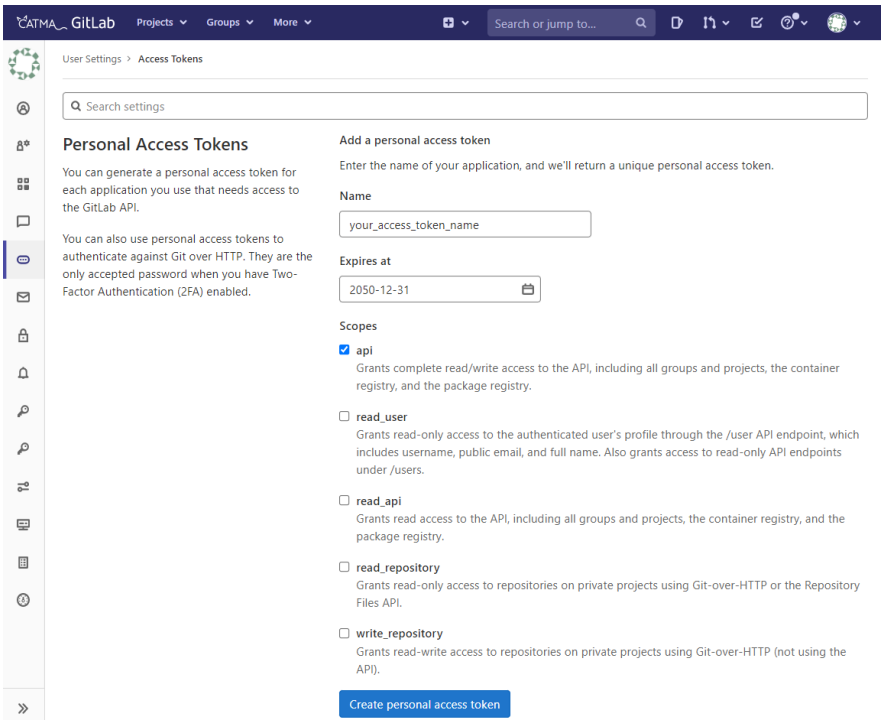

Abb. 8: API-Settings im CATMA-GitLab nutzen

Nutzen Sie dann den folgenden Code, um Ihr CATMA-Projekt zu laden.

```
your_access_token = ''
my_catma = Catma(gitlab_access_token=your_access_token)

# Um eine Liste Ihrer Projekte einzusehen, können Sie den Befehl nutzen:

my_catma.project_name_list
```

3. CATMA-Projekt klonen und laden

Mit den vorangehenden Schritten haben Sie die Verbindung zu Ihrem CATMA-Profil hergestellt. Diese Verbindung können Sie nutzen, um einzelne Projekte zu klonen, d.h. lokal auf Ihrem Rechner zu speichern. Diese lokal gespeicherten Daten können Sie mithilfe von GitMA laden und analysieren. Um ein Projekt zu klonen, brauchen Sie lediglich dessen Namen.
Um Ihr Projekt zu klonen und zu laden, passen Sie untenstehenden Code so an, dass zwischen den einfachen Anführungsstrichen, in denen jetzt „test_corpus“ steht, Ihr Projektname eingesetzt wird. Einen Ordner, in dem das Projekt gespeichert werden soll, können Sie in der Zeile darunter zwischen den einfachen Anführungsstrichen angeben, dort, wo jetzt „../projects/“ steht.

```
my_catma.load_project_from_gitlab(
project_name='',
backup_directory='../projects/'
)
```

Aufgabe 1: In welcher Spalte der Tabelle finden Sie Ihre Tagsets?

Sie haben hier eine Fehlermeldung erhalten? Beim ersten Laden kann es sein, dass der Zugang direkt vom Jupyter Notenbook aus nicht zustande kommt. Fahren Sie darum mit Punkt 5.1 fort und kehren Sie dann hierher zurück.

4. Projekt-Statistiken einsehen

Sie haben über das Access Token eine Verbindung zu Ihrem CATMA-Account hergestellt und ein Projekt geklont. Um sich einige grundsätzliche Statistiken Ihres Projektes in einer Tabelle anzusehen, führen Sie den folgenden Code aus. Geben Sie dazu zunächst den Namen Ihres Projektes zwischen den einfachen Anführungsstrichen ein und klicken Sie dann oben im Menü auf „Run“.

```
my_catma.project_dict[''].stats()
```

5. Zusätzliche Funktionen

Mithilfe des ersten Jupyter Notebooks dieser Lerneinheit können Sie die Auswertung Ihrer Annotationen mit GitMA vorbereiten. Um weitere Analysen durchführen zu können, fahren Sie bitte mit dem 2. Demo-Notebook „Annotationen auswerten mit GitMA 2“ dieser Einheit fort. Falls Sie Ihr Projekt lieber manuell über ein Terminal (vgl. Commandline) klonen möchten oder alle Ihre Projekte laden, so finden Sie hierunter noch zwei zusätzliche Zeilen Code, die Sie dafür nutzen können.

5.1 Projekt über das Terminal klonen

Indem Sie den folgenden Code ausführen, erhalten Sie eine Zeile Code in einfachen Anführungsstrichen. Kopieren Sie diesen (ohne Anführungszeichen) in die Zwischenablage. Gehen Sie dann zurück zu Anaconda und wechseln Sie dort im Menü auf „Environment“ und klicken Sie auf den Play-Button hinter „Base (root)“. Gehen Sie dann auf „Open Terminal“ und kopieren Sie den Code aus Ihrer Zwischenablage hier hinein. Drücken Sie nun „Enter“, so wird der Download Ihres CATMA-Projektes gestartet. Ist es der erste Download aus einem CATMA-Account, so werden Sie aufgefordert, zuerst Ihren Benutzernamen und dann Ihr Passwort einzugeben (Achtung: Beim Eingeben des Passworts werden die Buchstaben aus Sicherheitsgründen nicht angezeigt). Drücken Sie nach Passworteingabe auf „Enter“, so startet der Download. Haben Sie diesen Schritt ausgeführt, da Sie bei Punkt „3. CATMA Projekt klonen und laden“ eine Fehlermeldung erhalten haben, so können Sie bei folgenden Downloads ab jetzt einfach den Code unter 3. im ersten Notebook dieser Lerneinheit ausführen. Die Eingabe der Benutzerdaten muss nur einmal erfolgen.

```
my_catma.git_clone_command(project_name='')
```

5.2 Alle CATMA-Projekte eines Accounts laden

Den folgenden Code können Sie nutzen, um all Ihre CATMA-Projekte zu laden. Nutzen Sie diese Funktion **vorsichtig**, denn bei vielen und / oder großen Projekten kann das Laden sehr lange dauern.

```
my_catma.load_all_projects_from_gitlab
```

Annotationsdaten explorieren

Im zweiten Teil dieser Lerneinheit nutzen wir ein CATMA-Demo-Projekt, um einige generelle statistische Eigenschaften zu betrachten. Mithilfe des zweiten Notebooks lernen Sie außerdem einige grundlegende Möglichkeiten der Visualisierung (Horstmann und Stange 2024) Ihrer Annotationsdaten kennen. Sie lernen, unterschiedliche *Annotation Collections* zu visualisieren und wie Sie *Tags* und *Properties* (vgl. Property) grafisch darstellen lassen können. Um diese Funktionen mit eigenen Annotationsdaten ausführen zu können, müssen Sie diese mit dem Annotationstool CATMA erstellt haben. Wie Sie in CATMA annotieren, erfahren Sie in der forTEXT-Lerneinheit zur manuellen Annotation mit CATMA (Horstmann 2024).
Um eigene CATMA-Annotationsdaten auswerten zu können, müssen diese außerdem zuerst lokal auf Ihrem Computer gespeichert werden. Wie Sie das mit GitMA machen können, erfahren Sie im ersten Teil dieser Lerneinheit.

1. CATMA-Projekt laden

Ein CATMA-Projekt besteht aus Dokumenten, in denen Text abgelegt ist, Tagsets (vgl. Tagset), in denen die Annotationskategorien abgelegt sind und Collections, in denen die Positionen und Tags der Annotationen pro Text und Annotator*in abgelegt sind. Diese komplexe Struktur können Sie mit GitMA laden, indem Sie den Code in der folgenden Zeile ausführen.

```
from gitma import CatmaProject
```

Als nächstes müssen Sie spezifizieren, welches Projekt genau geladen werden soll und wo es im Ordnersystem Ihres Computers zu finden ist. Um das Demo-Projekt zu nutzen, müssen Sie in untenstehendem Code nichts verändern.
Wenn Sie eigene CATMA-Daten nutzen wollen, so ändern Sie in den einfachen Anführungsstrichen, in denen jetzt „../test/demo_project/“ steht, den Dateipfad ab, indem Sie den Pfad zum Ordner eingeben, in dem Ihr CATMA-Projekt abgelegt wurde. Haben Sie im ersten Teil dieser Lerneinheit einen eigenen Backup-Ordner angelegt, so geben Sie jetzt hier den Pfad zu genau diesem Ordner ein, also z.B. „../gitma_demo/catma-backup“. In die einfachen Anführungsstriche, in denen jetzt „test_corpus“ steht, geben Sie den Namen Ihres CATMA-Projektes ein.

```
my_project = CatmaProject(
    project_directory='../test/demo_project/',
    project_name='test_corpus'
    )
```

2. Generelle Statistiken aufrufen

Wenn Sie sich einen Überblick darüber verschaffen wollen, wie viele Annotationen in Ihrem Projekt sind, wer aus dem Team wie viele davon erstellt hat und welche Tags verwendet wurden, so können Sie sich dafür eine Tabelle anzeigen lassen. Nutzen Sie dazu den folgenden Code:

```
my_project.stats()
```

3. Einen Überblick über die Annotationen eines Projektes bekommen

Sie können Ihre Annotationen in einer interaktiven Visualisierung grafisch darstellen. Mit der hier gezeigten Funktion wird jedes Dokument einzeln angezeigt. Das heißt, wenn Sie zu zweit zwei Texte annotiert haben, so erhalten Sie zwei Grafiken. Da in unserem Demo-Projekt nur ein Text annotiert wurde, erhalten Sie nur eine

Grafik, wenn Sie dieses Projekt nutzen. Sie können wählen, wofür die farbliche Darstellung in der Visualisierung stehen soll. Im ersten Beispiel sind dies die *Annotation Collections*, im zweiten eine *Property* der Annotationen und im dritten die Annotator*innen.

```
my_project.plot_interactive()
```

Aufgabe 2: Vergleichen Sie die beiden Visualisierungen. Welche Textstelle wurde in einer *Annotation Collection* als *Stative Event* und in der anderen als *Process* annotiert?
Wenn Sie statt der *Annotation Collections* lieber die Verwendung einer *Property* farbig darstellen wollen, nutzen Sie dazu den folgenden Code:

```
my_project.plot_interactive(color_col='prop:intentional')
```

Aufgabe 3: Wie viele Annotationen weisen die *Property* „intentional“ nicht auf?
Statt der Annotator*innen und der *Annotation Collections* können Sie auch die *Tags* visualisieren. Nutzen Sie dazu die folgende Code-Zeile:

```
my_project.plot_interactive(color_col='annotator')
```

Aufgabe 4: Aus der Tabelle unter „2. Generelle Statistiken aufrufen“ können Sie ablesen, dass beide Annotatoren jeweils sechs Annotationen gemacht haben. Wie erklären Sie sich, dass hier nur insgesamt acht Annotationen angezeigt werden?

4. Spezifische Annotation Collections visualisieren

Wenn Sie nur eine bestimmte *Annotation Collection*, d.h. nur die Annotationen eines einzigen Team-Mitglieds, visualisieren möchten, so können Sie dazu die untenstehende Funktion nutzen. Dargestellt sind auf der y-Achse die *Tags*, also die Annotationskategorien, und auf der x-Achse die Positionen im Text (in Zeichen, d.h., Position 500 bedeutet nicht Seite 500, sondern das 500. Zeichen im Text). Die Annotationen sind nach Tags (Farbe) und Umfang (Größe) einmal als Punkte im Koordinatensystem und einmal oberhalb desselben visualisiert. Oben sehen Sie, wie die Annotationen sich über den Text erstrecken und einander auch überlappen. Sowohl im Koordinatensystem als auch darüber können Sie die Maus-Hover-Funktion nutzen, um mehr Informationen zu erhalten. Führen Sie Ihre Maus z.B. über die zweite Annotation eines „stative_event“ im Koordinatensystem, so erfahren sie u.a. die Größe der Annotation und auch den genauen Text, der annotiert worden ist.
Wenn Sie eigene CATMA-Daten für Visualisierungen dieser Art nutzen, so ersetzen Sie „ac_2“ in den einfachen Anführungszeichen durch den Namen der *Annotation Collection*, die Sie untersuchen möchten.

```
my_project.ac_dict['ac_2'].plot_annotations()
```

Sie können die Visualisierung anpassen, sodass sie anzeigt, ob Annotationen bestimmte *Properties* aufweisen. In unserem Beispielprojekt gibt es z.B. die Eigenschaft „mental“ für die annotierten Events. Mit dem folgenden Code können Sie sichtbar machen, welche der annotierten Events auf mentale Ereignisse referieren.

```
my_project.ac_dict['ac_2'].plot_annotations(y_axis='prop:mental')
```

Wenn Sie nachvollziehen möchten, ob die Annotator*innen *Properties* unterschiedlich verwendet haben, so können Sie mit folgendem Code die Annotator*innen auf der y-Achse und die Annotationen auf der x-Achse anzeigen lassen. Die Farbe zeigt die Verwendung der *Property* an (hier: Unvorhersehbarkeit der annotierten Events). In unserem Beispielprojekt hat keiner der Annotatoren ein Event als unvorhersehbar klassifiziert.

```
my_project.ac_dict['ac_2'].plot_annotations(
    y_axis='annotator',
    color_prop='prop:unpredictable'
    )
```

Aufgabe 5: Statische oder Prozess-Events: Mit welcher Annotationskategorie wurde die längste Textpassage annotiert und wie viele Zeichen umfasst diese?

5. Kookurrenznetzwerke von Annotationen darstellen

Sie können Ihre Annotationen als Netzwerke (mehr zur Methode der digitalen Netzwerkanalyse sieh Schumacher (2024b)) darstellen, indem Sie z.B. immer die am nächsten zueinanderstehenden Annotationskategorien berechnen lassen. Auf diese Weise bekommen Sie einen übergeordneten Blick auf Ihre Annotationskategorien. Sie können den folgenden Code ohne Anpassung ausführen, um mit unseren Beispieldaten ein sehr einfaches Netzwerk darzustellen. Wenn Sie mit eigenen Projektdaten arbeiten, so tauschen Sie „ac_1“ gegen den Namen einer Ihrer *Annotation Collections* aus.

```
my_project.ac_dict['ac_1'].cooccurrence_network()
```

Sie können das Netzwerk anpassen, indem Sie folgende Parameter hinzufügen und ggf. abändern:

- character_distance: Textspanne, in der sich zwei als kookurrent interpretierte Annotationen max. befinden können (Default Wert sind hier 100 Zeichen)
- included_tags: Tags, die in die Netzwerkerstellung einbezogen werden sollen (in Form einer Liste)
- excluded_tags: Tags, die nicht in die Netzwerkerstellung einbezogen werden sollen (in Form einer Liste oder „None“, wenn keine Tags ausgeschlossen werden sollen)

Da unser bisher genutztes Demo-Projekt zu wenig Daten beinhaltet, um ein derart modifiziertes Netzwerk abzubilden, haben wir einen anderen Datensatz als Beispiel aufbereitet.
Wenn Sie mit eigenen Daten arbeiten, passen Sie wieder den Namen der *Annotation Collection* (hier: ac_1) an. Die übrigen Werte können Sie unverändert stehen lassen oder abändern, je nach Bedarf. Wenn Sie die Listen der berücksichtigten und ausgeschlossenen Tags erweitern wollen, vergessen Sie nicht, jeden hinzugefügten Tag in einfache Anführungsstriche zu setzen und mit einem Komma von dem vorigen zu trennen.

```
my_project.ac_dict['ac_1'].cooccurrence_network(
    character_distance=50,
    included_tags=['process', 'stative_event'],
    excluded_tags=None
    )
```

6. Annotation Collections als Pandas DataFrame darstellen

Mit der nächsten Funktion können Sie die Annotationen einer *Annotation Collection* und deren Kontexte in einer Tabelle darstellen. Wie bei den Visualisierungen im vorigen Abschnitt, können Sie entweder mit dem Demo-Projekt arbeiten und den Code unverändert ausführen oder mit eigenen CATMA-Daten arbeiten. Wenn Sie mit eigenen CATMA-Daten arbeiten, ersetzen Sie einfach wieder „ac_2“ durch den Namen einer Ihrer *Annotation Collections*.

```
my_project.ac_dict['ac_2'].df
```

Pandas (McKinney u. a. 2010) ist eine vielseitige und populäre Python Library um quantitative Analysen größerer Datenmengen durchzuführen. Als pandas DataFrame können Annotation Collections auch in unterschiedlichen Datenformaten gespeichert werden; zum Beispiel tabellarisch als CSV-Dateien:

```
my_project.ac_dict['ac_2'].df.to_csv('annotations.csv')
```

oder als JSON Dateien:

```
my_project.ac_dict['ac_2'].df.to_json('annotations.json')
```

In dieser Form lassen sich CATMA-Annotationen auch gut nachnutzbar veröffentlichen.

6.1 Tags analysieren

Möchten Sie herausfinden, wie viele Annotationen jeweils mit einem Tag gemacht wurden, wie viel Text jeweils mit einem Tag belegt wurde, wie die durchschnittliche Länge einer Annotation mit einem Tag ist und welche Wörter darin am häufigsten vorkommen? All das können Sie mit der nächsten Funktion machen. Wie gehabt, arbeiten Sie mit dem Demo-Projekt, wenn Sie den Code unverändert lassen. Wenn Sie mit eigenen Daten arbeiten, fügen Sie wieder den Namen Ihrer eigenen *Annotation Collection* ein. Sie können außerdem die Anzahl der häufigsten Tokens (vgl. Type/Token) eines Tags, die aufgelistet werden soll, anpassen. Um dies zu tun, ersetzen Sie die 5 beim „ranking“ durch eine andere Zahl.

```
my_project.ac_dict['ac_2'].tag_stats(ranking=5)
```

Wenn Sie *Properties* verwendet haben, so können Sie die Funktion auch so anwenden, dass Ihnen nur Annotationen aufgelistet werden, bei denen eine bestimmte *Property* verwendet wurde. In untenstehendem Beispiel werden nur Annotationen von Events berücksichtigt, bei der die *Property* „mental“ genutzt wurde. In den Reihen zeigen sich die unterschiedlichen Werte der *Property*, die vergeben wurden (hier „no“ und „yes“). Beim Ranking werden die 3 häufigsten Wörter berücksichtigt und es werden Stopwörter (vgl. Stoppwortliste) ausgeschlossen. Außerdem können Sie weitere Wörter ergänzen. Setzen Sie dabei jedes Wort in einfache Anführungsstriche und trennen Sie die Wörter mit einem Komma. Wie bei den vorherigen Code-Beispielen, können Sie auch hier wieder mit eigenen Daten arbeiten und „ac_2“ durch den Namen einer Ihrer *Annotation Collections* ersetzen.

```
my_project.ac_dict['ac_2'].tag_stats(tag_col='prop:mental', ranking=3,
    ↪ stopwords=['in', 'im'])
```

Wenn Sie möchten, dass in den Reihen nicht die Property Values, sondern die Annotator*innen angegeben werden, so nutzen Sie den folgenden Code. Es werden im Beispiel die drei häufigsten Wörter in die Analyse einbezogen. Auch hier kann bei Bedarf „ac_2“ wieder durch den Namen einer eigenen *Annotation Collection* ersetzt werden.

```
my_project.ac_dict['ac_2'].tag_stats(tag_col='annotator', ranking=3)
```

6.2 *Properties* analysieren

Wenn Sie die Nutzung aller *Properties* und deren Wertung analysieren wollen, so können Sie das mit der folgenden Funktion tun. Wenn Sie mit eigenen Daten arbeiten, ersetzen Sie wie gehabt „ac_2“.

```
my_project.ac_dict['ac_2'].property_stats()
```

Aufgabe 6: Wie viele Events wurden als *intentional* klassifiziert, wie viele als *nicht-intentional* und bei wie vielen Annotationen wurde diese *Property* nicht mit einem Wert versehen?

Ein Projekt laden, sich einen Überblick darüber verschaffen, *Annotation Collections* visualisieren und Tags und deren *Properties* analysieren – all das können Sie nun bereits mithilfe von GitMA tun. Im nächsten Teil dieser Lerneinheit zeigen wir Ihnen, wie Sie mit GitMA eine Goldstandard-Annotation erstellen und in Ihr Projekt in CATMA hochladen können. Um mit dem dritten Teil der Lerneinheit zu beginnen, öffnen Sie bitte das Jupyter Notebook mit dem Namen „Annotationen auswerten mit GitMA 3“.

Goldstandard-Annotationen automatisch erstellen mit GitMA

Wir wenden uns nun der Erstellung eines Goldstandards für Annotationen zu, die Sie in CATMA erstellt haben. Diese Daten erstellen Sie zunächst lokal auf Ihrem Computer, können Sie dann aber zurück in die CATMA-Webapplikation laden.
Der dritte Teil dieser Lerneinheit nutzt wieder ein von uns erstelltes Beispielprojekt. Wie Sie eigene CATMA-Daten auf Ihren Computer herunterladen können, zeigen wir Ihnen im ersten Teil, dem Notebook mit dem Dateinamen „Annotationen auswerten mit GitMA 1“.

1. Einführung in die Erstellung eines Goldstandards mit GitMA

Um Goldannotationen zu erstellen, kann mit GitMA eine Kopie aller übereinstimmenden Annotationen von zwei Annotator*innen angefertigt werden. Diese übereinstimmenden Annotationen werden in einer (neuen) *Gold Annotation Collection* abgelegt. Diese neue *Annotation Collection* können Sie über Git wieder in CATMA hochladen, wo Sie sie dann wie gewohnt bearbeiten können, d.h., Sie können hier Annotationen hinzufügen und die automatisch erstellten Gold Annotationen überarbeiten.
In CATMA können Sie ganz frei und undogmatisch annotieren, was dazu führt, dass Annotationsspannen stark variieren können. Bei der Erstellung eines Goldstandards mit GitMA kann darum flexibel angepasst werden, was als Übereinstimmung gewertet werden soll.
Um die erstellten Annotationsdaten automatisch in die CATMA-Webapp hochladen zu können, erstellen Sie in Ihrem CATMA-Projekt eine neue *Annotation Collection*, z.B. mit dem Namen „Gold_Annotation“. Kehren Sie zu dieser Lerneinheit zurück, wenn das erledigt ist.

2. CATMA-Projekt laden

Wenn Sie Teil 2 dieser Lerneinheit bereits durchgeführt haben, wissen Sie schon, dass Sie das CATMA-Projekt, mit dem Sie arbeiten wollen, immer zuerst laden müssen. Um das zu tun, nutzen Sie untenstehenden Code.

```
from gitma import CatmaProject
```

Danach können Sie das CATMA-Demo-Projekt laden. Wenn Sie mit eigenen Daten arbeiten wollen, so ersetzen Sie in der zweiten Code-Zeile „../test/demo_project“ durch den Dateipfad, der zu Ihrem CATMA-Projekt führt (Achtung: Sie müssen hier nur den Dateipfad zu dem übergeordneten Ordner einfügen, in dem Ihr Projekt liegt. Das Projekt selbst hat einen Namen, der diesem ähnelt: CATMA_374270D5-006D-498D-BDB4-12D068DEFDC0_Projektname_root. Sie müssen hier aber eine Ebene darüber in Ihrem Ordnersystem angeben.). Geben Sie außerdem den Projektnamen (nicht den Namen des Projektordners) dort ein, wo jetzt „test_corpus“ steht.

```
my_project = CatmaProject(
     project_directory='../test/demo_project/',
     project_name='test_corpus'
     )
```

3. Goldannotationen automatisch erstellen

Um die Erstellung von Goldstandard-Annotation zu unterstützen, müssen Sie im ersten Schritt zwei Annotationssammlungen vergleichen und im zweiten nur diejenigen Annotationen behalten, die übereinstimmend annotiert wurden. Im Demo-Projekt dieser Lerneinheit gibt es zwei *Annotation Collections*, die „ac_1“ und „ac_2“ genannt wurden. Außerdem gibt es die leere *Annotation Collection* namens „gold_annotations“. Als Übereinstimmung werden hier Annotationen gewertet, die sich zu mindestens 95% überlappen. Eine Annotation kann also z.B. etwas länger sein als die andere oder einen leicht anderen Start- und/oder Endpunkt haben. Die genutzten Annotationskategorien (Tags) müssen die gleichen sein und die *Properties* müssen gleich vergeben worden sein. Um die Funktion mit den Demo-Daten auszuprobieren, müssen Sie den Code nicht anpassen.
Wenn Sie mit eigenen Daten arbeiten, so müssen Sie die Namen der *Annotation Collections* durch diejenigen ersetzen, die Sie in Ihrem Projekt verwenden. Achten Sie dabei auf die exakte Schreibweise! Sie können bestimmte Tags bei der Erstellung einer Goldstandard-Annotation ausschließen. Geben Sie dazu die Namen der Tags in die eckigen Klammern hinter „excluded_tags=“ ein. Setzen Sie jeden Tagnamen in Anführungsstriche und trennen Sie die einzelnen Tags mit Kommas voneinander. Wenn Sie die Definition übereinstimmender Annotationen anpassen möchten, geben Sie statt der 0,95 bei „min_overlap“ einen anderen Wert ein. Möglich sind Werte von 0-1, wobei 0 für keinerlei Überlappung steht und darum hier kein sinnvoller Wert wäre und 1 für eine 100%-ige Überlappung. Möchten Sie auch Annotationen als übereinstimmend werten, bei denen zwar dieselbe Textpassage annotiert wurde, nicht aber mit demselben Tag, so geben Sie statt „same_tag=True“ „same_tag=False“ ein. Bei „property_values“ können Sie statt „matching“ auch „none“ eingeben. In diesem Fall werden *Property Values* nicht in die Erstellung der Goldstandard-Annotationen einbezogen. Wenn Sie eine automatisch erstellte Goldstandard-Annotation über GitLab in Ihr CATMA-Projekt hochladen wollen (dies geht nur mit eigenen CATMA-Daten), so geben Sie „True“ bei „push_to_gitlab“ ein.

```
my_project.create_gold_annotations(
     ac_1_name='ac_1',
     ac_2_name='ac_2',
     gold_ac_name='gold_annotation',
     excluded_tags=[],
     min_overlap=0.95,
     same_tag=True,
     property_values='matching',
     push_to_gitlab=False
     )
```

Aufgabe 7: Wie viele Annotationen wurden jeweils in den *Annotation Collections* gefunden und wie viele davon stimmen nicht zu 95% überein?

Wenn Sie mit eigenen Daten arbeiten und die Goldstandard-Annotationen automatisch per GitLab in Ihren CATMA-Account hochgeladen haben, so müssen Sie Ihr CATMA-Projekt noch synchronisieren, damit die Änderungen dort sichtbar werden. Loggen Sie sich dazu in Ihren CATMA-Account ein und gehen Sie in Ihr Projekt. Klicken Sie rechts auf das Drei-Punkte-Menü und dann auf „Synchronize with the team“ (vgl. Abb. 9). Ihre Änderungen werden sichtbar sein, sobald der Vorgang abgeschlossen ist.

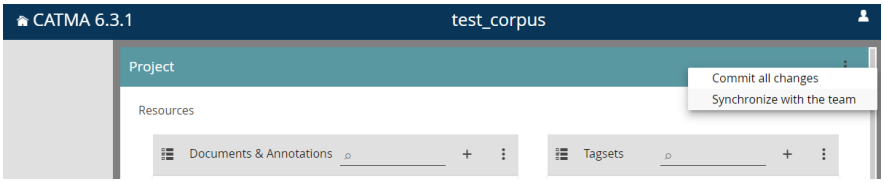

Abb. 9: CATMA-Projekt synchronisieren

Sie wissen nun, wie Sie einen Goldstandard für übereinstimmende Annotationen errechnen und erstellen lassen können und diesen auch zurück in Ihr CATMA-Projekt laden. Vielleicht möchten Sie die Übereinstimmungen bei der Annotation in Ihrem Projekt aber viel genauer analysieren. Dabei helfen Berechnungen des Inter-Annotator-Agreements. Wie Sie diese mit CATMA-Daten durchführen, lernen Sie im vierten Teil dieser Lerneinheit. Öffnen Sie dazu nun das Notebook mit dem Titel „Annotationen auswerten mit GitMA 4“.

Inter-Annotator-Agreement berechnen

Wenn Sie in CATMA mit Ihrem Team kollaborativ annotiert haben, so interessiert es Sie vielleicht, wie hoch die Übereinstimmung der Annotator*innen miteinander ist. Eine solche Übereinstimmung (auch Inter-Annotator-Agreement oder kurz IAA genannt) kann auf unterschiedliche Weisen berechnet werden. GitMA bietet insgesamt vier Varianten der Berechnung eines IAAs für zwei oder mehr Annotator*innen an: *Scott's pi*, *Cohen's kappa*, *Krippendorf's alpha* (Artstein und Poesio 2008) und das *Gamma*-Agreement (Mathet, Widlöcher und Métivier 2015). Die ersten drei IAA-Varianten können mit dem Python Package *Natural Language Toolkit* (Bird, Klein und Loper 2009) berechnet werden, die vierte mit dem Package *pygamma-agreement* (Titeux und Riad 2021). Bei der Berechnung von Gamma wird mit einbezogen, dass die Einheiten, die annotiert werden, von den Annotator*innen individuell gewählt werden können. In CATMA kann gänzlich individuell annotiert werden, d.h., es können Buchstabenkombinationen, Wörter, Phrasen, Satzteile, Sätze, freie Passagen oder Absätze usw. annotiert werden und das auch noch auf verschiedene Weisen überlappend (Horstmann 2024). Darum kann es entscheidend sein, bei der Berechnung des IAA das sogenannte *Unitizing*, also die Variabilität der Units / Annotationseinheiten, zu berücksichtigen. Leider gibt es für die Berechnung des IAA keine Standard-Variante, die für alle Arten von Annotationsprojekten empfohlen werden kann (Pagel u. a. 2020). Stattdessen muss individuell entschieden werden, welcher Wert genau berechnet werden soll. Wenn Sie die Übereinstimmung zwischen Annotator*innen genauer betrachten wollen, so öffnen Sie dazu das vierte Jupyter Notebook zu dieser Lerneinheit mit dem Namen „Annotationen auswerten mit GitMA 4.ipynb" und folgen Sie den Anleitungen im Notebook.
Wir empfehlen, jeweils eine *Annotation Collection* pro Annotator*in und Dokument anzulegen. Am besten wird jede *Annotation Collection* so benannt, dass Titel, Annotationsaufgabe und Annotator*in enthalten sind.

Zum Beispiel: Robinson_Crueso-narrativer_Raum-Mareike

1. CATMA-Projekt laden

Für diese Lerneinheit haben wir ein Beispielprojekt vorbereitet, das Annotationen von zwei Annotatoren zu Kafkas *Urteil* enthält. Wenn Sie mit eigenen Daten arbeiten wollen, so können Sie diese aus Ihrem CATMA-Account herunterladen. Wie Sie das tun können, erfahren Sie im ersten Notebook („Annotationen auswerten mit GitMA 1") zu dieser Lerneinheit. Wenn Sie das Demo-Projekt nutzen möchten, so können Sie den Code unverändert übernehmen.
Wenn Sie mit eigenen Daten arbeiten möchten, müssen Sie den Code leicht anpassen. Schreiben Sie in die einfachen Anführungsstriche hinter „project_name", in denen jetzt „test_corpus" steht, den Namen Ihres CATMA-Projektes. Hinter „project_directory" müssen Sie den Dateipfad angeben, der zu dem Ordner auf Ihrem Computer führt, in dem das CATMA-Projekt abgelegt wurde. Ersetzen Sie „../test/demo_project/" durch den Dateipfad, den Sie im ersten Notebook dieser Lerneinheit als „backup_directory" angegeben haben.

```
from gitma import CatmaProject

my_project = CatmaProject(
    project_name='test_corpus',
    project_directory='../test/demo_project/'
    )
```

2. Einführung in die Funktion get_iaa()

Im Demo-Projekt für diese Lerneinheit gibt es einen Text mit insgesamt drei *Annotation Collections*. Das Inter-Annotator-Agreement wird für zwei dieser *Annotation Collections* errechnet (die *Annotation Collections* wurden als „ac_1" und „ac_2" benannt).
Mit der Funktion get_iaa() wird für jede Annotation in einer *Annotation Collection* die am besten passende Annotation in einer zweiten Collection gesucht. Der Vergleich basiert auf der annotierten Textpassage, also auf der Annotationsspanne. Die folgenden Grafiken verdeutlichen, wie Annotationen ausfindig gemacht werden, die zueinander passen (vgl. Abb 10).

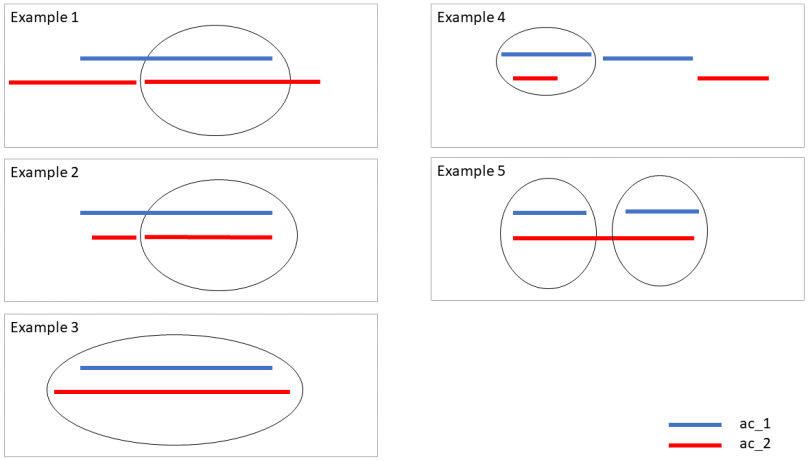

Abb. 10: Unterschiedlich gut zueinander passende Annotationen

Von allen Annotationen werden bei den IAA-Varianten, die mit dieser Funktion berechnet werden können, immer nur die berücksichtigt, die am besten passt.

2.1 Allgemeines Beispiel

Wenn Sie zunächst einmal die Annotationsspannen in zwei *Annotation Collections* vergleichen wollen, so können Sie das mithilfe einer Visualisierung tun. Die Annotationsspannen werden dabei anhand ihrer Anfangspunkte verglichen. Wenn Sie mit dem Demo-Projekt arbeiten, so nutzen Sie dazu einfach untenstehenden Code.
Wenn Sie mit eigenen Daten arbeiten, so ersetzen Sie in der eckigen Klammer die Namen der *Annotation Collections* aus dem Demo-Projekt durch die aus Ihrem eigenen CATMA-Projekt.

```
my_project.compare_annotation_collections(annotation_collections=['ac_1', 'ac_2
    ↪ '])
```

Im nächsten Schritt berechnen wir nun die Übereinstimmung für alle passenden Annotationen. Klicken Sie dazu in die nächste Box und dann oben auf „Run“. Wenn Sie mit eigenen Annotationsdaten arbeiten, vergessen Sie nicht, die Namen der *Annotation Collections* durch die Ihrer eigenen zu ersetzen.

```
my_project.get_iaa(ac1_name='ac_1',ac2_name='ac_2')
```

Aufgabe 8: Wie Sie sehen, gibt Ihnen diese Funktion drei Varianten des IAAs aus. Welche Informationen erhalten Sie darüber hinaus zur Übereinstimmung der Annotationen? Vergleichen Sie die ausgegebene Tabelle mit der Visualisierung, die Sie vorhin erstellt haben und den Beispielen für passende Annotationen. Für welche drei statischen Events gibt es passende Annotationen in der zweiten *Annotation Collection* und welchem Beispiel für passende Annotation entspricht das?

2.2 Nach Annotationskategorien filtern

Wenn Sie nicht für alle Annotationskategorien in Ihren *Annotation Collections* die Übereinstimmung der Annotationen berechnen wollen, sondern nur für ausgewählte Tags, so können Sie das mit dem folgenden Code tun. Möchten Sie das Demo-Projekt nutzen, so können Sie wie gewohnt den Code unverändert ausführen.
Wenn Sie eigene Daten nutzen wollen, tauschen Sie wieder die Namen der *Annotation Collections* aus. Ändern Sie dann den Namen des Tags in den eckigen Klammern bei „tag_filter“ und tragen Sie hier den Namen einer Ihrer eigenen Annotationskategorien ein. Wenn Sie weitere Tags berücksichtigen möchten, so ergänzen Sie in den eckigen Klammern einfach weitere Namen von Tags aus Ihren *Annotation Collections*. Setzen Sie dabei jeden Tagnamen in einfache Anführungsstriche und setzen Sie Kommas dazwischen.

```
my_project.get_iaa(
    ac1_name='ac_1',
    ac2_name='ac_2',
    tag_filter=['process']
    )
```

Aufgabe 9: In der Abfrage berücksichtigen Sie nur eine Annotationskategorie (*process*). Wie erklären Sie sich, dass hier trotzdem in der Tabelle auch die andere Annotationskategorie auftaucht?

Tipp: Berücksichtigen Sie bei Ihrer Antwort auch die Visualisierung und die grafischen Beispiele für passende Annotationen!
Sie können diesen Effekt herausnehmen, indem Sie nicht nur von einer *Annotation Collection* als Basis ausgehen, sondern den Filter, der nur bestimmte Tags in die Berechnungen einbezieht, auf beide *Annotation Collections* ausweiten. Nutzen Sie dazu den folgenden Code wie gehabt unverändert, wenn Sie das Demo-Projekt nutzen wollen, oder passen Sie ihn wie oben an, wenn Sie eigene Daten nutzen.

```
my_project.get_iaa(
     ac1_name='ac_1',
     ac2_name='ac_2',
     tag_filter=['process'],
     filter_both_ac=True
     )
```

Wie Sie sehen, wird nun in der vierten Zeile des Ergebnisses angezeigt, dass für eine Annotation keine passende in der zweiten *Annotation Collection* gefunden wurde. Trotzdem sind die Übereinstimmungswerte dieselben wie in der obigen Berechnung. Das liegt daran, dass es in unserem Demo-Projekt nur zwei unterschiedliche Tags gibt. Wenn Sie mit eigenen Daten arbeiten, können die Ergebnisse ganz anders aussehen.

2.3 Annotationen mit *Properties* vergleichen

In CATMA können Sie für jede Annotationskategorie eine oder mehrere Eigenschaften (*Properties*) angeben, die zwei oder mehr Ausprägungen (*Values*) haben können. Tags sind also nur eine Ebene von CATMA-Annotationen und natürlich können Sie auch *Properties* und *Values* in die Berechnung des Inter-Annotator-Agreements einbeziehen. In unserem Beispielprojekt gibt es die *Property* „mental“, die immer dann verwendet wird, wenn ein im Text dargestelltes Ereignis nicht tatsächlich (innerhalb der erzählten Welt) stattfindet, sondern nur in der Vorstellung einer Figur (mental event). Auch hier erstellen wir zunächst wieder eine Visualisierung, um uns die Übereinstimmung grafisch vor Augen zu führen. Um dies mit den Demo-Daten zu tun, müssen Sie wieder nichts anpassen.
Wenn Sie eigene Daten nutzen, so ändern Sie wieder wie gehabt die Namen der *Annotation Collections* in den eckigen Klammern. Damit die Nutzung der *Properties* farblich dargestellt wird, geben Sie bei „color_col“ in die einfachen Anführungsstriche den Namen der *Property* ein, deren Nutzung Sie analysieren wollen.

```
my_project.compare_annotation_collections(
     annotation_collections=['ac_1', 'ac_2'],
     color_col='prop:mental'
     )
```

Um nun die mathematische Berechnung des Agreements durchzuführen, nutzen Sie einen sogenannten „Level“-Parameter. Um die Funktion mit den Beispieldaten auszuführen, können Sie einfach wieder den unveränderten Code nutzen.
Möchten Sie die Berechnung mit eigenen Daten durchführen, passen Sie auch hier wieder die Namen der *Annotation Collections* an. Ersetzen Sie dann noch „mental“ durch den Namen einer *Property* aus Ihrer *Annotation Collection*.

```
my_project.get_iaa(
     ac1_name='ac_1',
     ac2_name='ac_2',
     level='prop:mental'
     )
```

An diesem Beispiel zeigt sich sehr gut, dass diese Art, die Übereinstimmung zu berechnen, Uneinigkeiten außer Acht lässt, sofern diese nicht die am besten zueinander passenden Annotationen betreffen. Bei der letzten Annotation aus der ersten *Annotation Collection* wird die nicht-übereinstimmende Annotation aus der zweiten Collection nicht berücksichtigt, weil es sich dabei nicht um die am besten zu der Annotation aus „ac_1“ passende Annotation handelt. Wenn also unterschiedliche Annotationsspannen eine große Rolle spielen, so empfehlen wir die Berechnung des Gamma-Agreements.

3. Das Gamma-Agreement berechnen

Um das Gamma-Agreement zu berechnen, müssen Sie zusätzlich zu den *Annotation Collections* noch 5 weitere Parameter definieren:

- *alpha*: Koeffizient zur Gewichtung der Unähnlichkeit von Annotationen im Hinblick auf deren Position (default-Wert ist 1)
- *beta*: Koeffizient zur Gewichtung der Unähnlichkeit von Annotationen im Hinblick auf deren Annotationskategorie (default-Wert ist 1)
- *delta_empty*: Bewertung der Unähnlichkeit im Hinblick auf Leerstellen, d.h., wenn eine Textpassage von einem Annotator / einer Annotatorin annotiert wurde und vom / von der anderen nicht (default-Wert ist 1)
- *n_samples*: Anzahl der zufällig gewählten Annotationsspannenverteilungen, die genutzt werden, um einen erwarteten Gamma-Wert zu berechnen (im Beispiel wird die tatsächliche Verteilung der Annotationsspannen mit 30 Samples zufälliger Annotationsspannenverteilung verglichen.)
- *akzeptierte Fehleranfälligkeit* bei der Berechnung des erwarteten Gamma-Wertes (kann als „high“=1%, „medium“=2% oder „low“=5% spezifiziert werden)

Weitere Erklärungen zur Berechnung des Gamma-Agreements finden Sie in der Dokumentation von *pygamma* und bei Mathet, Widlöcher und Métivier (2015). Auch hier können Sie wieder unsere Demo-Daten für die Berechnung nutzen. Dafür müssen Sie den Code unverändert ausführen. Wenn Sie mit eigenen Daten arbeiten, passen Sie auch hier wieder die Namen der *Annotation Collections* so an, dass Sie denen entsprechen, die Sie in die Berechnung einbeziehen wollen.

```
my_project.gamma_agreement(
    annotation_collections=['ac_1', 'ac_2'],
    alpha=1,
    beta=1,
    delta_empty=1,
    n_samples=30,
    precision_level=0.01
    )
```

Wenn Sie mit weiteren, hier nicht aufgeführten Varianten das Inter-Annotator-Agreement direkt mit dem *pygamma* Package berechnen wollen, so können Sie dafür Rohdaten aus Ihrem CATMA-Projekt generieren. Mit dem unten stehenden Code können Sie diese als CSV-Tabelle speichern.
Nutzen Sie auch hier unser Demo-Projekt, indem Sie den nächsten Code unverändert ausführen. Passen Sie Folgendes an, um mit eigenen Daten zu arbeiten: Die Namen der *Annotation Collections* (wie gehabt) und den Dateipfad, unter dem die Tabelle gespeichert werden soll. Ändern Sie dazu die Angabe „../test/pygamma_table.csv“ so ab, dass hier der Pfad steht, unter dem Ihre Version der Tabelle in der Ordnerstruktur Ihres Rechners abgelegt werden soll.

```
pygamma_df = my_project.pygamma_table(
    annotation_collections=['ac_1', 'ac_2']
    )
    pygamma_df.to_csv('../test/pygamma_table.csv', index=False, header=False)
    pygamma_df.head(5)
```

4. Lösungen zu den Beispielaufgaben

Aufgabe 1: In welcher Spalte der Tabelle finden Sie Ihre Tagsets?

In der vierten Spalte der Tabelle finden Sie Ihre Tags. Die einzelnen Tags sind in geschweiften Klammern dargestellt, sodass sie zu Ihren Tagsets zusammengefasst sind.

Aufgabe 2: Vergleichen Sie die beiden Visualisierungen. Welche Textstelle wurde in einer *Annotation Collection* als *Stative Event* und in der anderen als *Process* annotiert?

Die Textstelle „Er hatte gerade einen Brief an einen sich im Ausland befindenden Jugendfreund gesendet“ wurde in der *Annotation Collection* mit der ID 1 als *Process* annotiert, in der *Annotation Collection* mit der ID 3 als *Stative Event*.

Aufgabe 3: Wie viele Annotationen weisen die *Property* „intentional“ nicht auf?

Vier Annotationen werden mit „nan“ angezeigt, was für „Not a number“ steht und in diesem Fall bedeutet, dass die *Property* nicht vergeben wurde, also auch keinen Wert hat.

Aufgabe 4: Aus der Tabelle unter „2. Generelle Statistiken aufrufen“ können Sie ablesen, dass beide Annotatoren jeweils sechs Annotationen gemacht haben. Wie erklären Sie sich, dass hier nur insgesamt acht Annotationen angezeigt werden?

Die Annotationen der Annotator*innen können auf denselben Textstellen liegen. In diesem Fall werden die Annotationen der ersten *Annotation Collection* angezeigt. Nur abweichende Annotationen bekommen einen eigenen Punkt in der Visualisierung und werden auch farblich entsprechend dargestellt.

Aufgabe 5: Statische oder Prozess-Events: Mit welcher Annotationskategorie wurde die längste Textpassage annotiert und wie viele Zeichen umfasst diese?

Mit der Annotationskategorie „Stative Event“ wurde eine Passage von insgesamt 233 Zeichen annotiert. Dies ist die längste annotierte Passage im Korpus des Demo-Projektes.

Aufgabe 6: Wie viele *Events* wurden als *intentional* klassifiziert, wie viele als *nicht-intentional* und bei wie vielen Annotationen wurde diese *Property* nicht mit einem Wert versehen?

Von den insgesamt sechs Annotationen wurde eine mit der *Property* „intentional“ und dem Wert/Value „yes“ versehen, eine weitere mit „intentional“ und „no“. Bei vier Annotationen wurde die *Property* „intentional“ nicht vergeben.

Aufgabe 7: Wie viele Annotationen wurden jeweils in den *Annotation Collections* gefunden und wie viele davon stimmen nicht zu 95% überein?

In jeder *Annotation Collection* wurden sechs Annotationen gefunden. Zwei davon stimmen zu mindestens 95% überein. Insgesamt acht Annotationen stimmen zu weniger als 95% überein.

Aufgabe 8: Wie Sie sehen, gibt Ihnen diese Funktion drei Varianten des IAAs aus. Welche Informationen erhalten Sie darüber hinaus zur Übereinstimmung der Annotationen? Vergleichen Sie die ausgegebene Tabelle mit der Visualisierung, die Sie vorhin erstellt haben und den Beispielen für passende Annotationen. Für welche drei statischen Events gibt es passende Annotationen in der zweiten *Annotation Collection* und welchem Beispiel für passende Annotationen entspricht das?

Es wird angezeigt, wie viele zusammenpassende Annotationen in den *Annotation Collections* gefunden wurden (im Demo-Projekt 6). Außerdem erfährt man noch, wie hoch die durchschnittliche Überlappung ist (im Demo-Projekt rund 83%) und wie viele Annotationen es gibt, für die keine passende Entsprechung in der anderen *Annotation Collection* gefunden wurde (im Demo-Projekt 0). Für die ersten drei Annotationen statischer Events in der Annotation Collection „ac_1“ gibt es passende Annotationen in „ac_2“. Es handelt sich dabei um die ersten beiden Annotationen aus dieser zweiten *Annotation Collection*. Die ersten beiden Annotationen umfassen eine ähnliche Textspanne, wie in Beispiel 4 gezeigt. Die zweite und dritte Annotation der zweiten *Annotation Collection* fallen in die Spanne der zweiten Annotation in der zweiten *Annotation Collection*. Dies entspricht Beispiel 5 für passende Annotationen.

Aufgabe 9: In der Abfrage berücksichtigen Sie nur eine Annotationskategorie (*process*). Wie erklären Sie sich, dass hier trotzdem in der Tabelle auch die andere Annotationskategorie auftaucht? Tipp: Berücksichtigen Sie bei Ihrer Antwort auch die Visualisierung und die grafischen Beispiele für passende Annotationen!

Die Berechnung geht immer von einer *Annotation Collection* (im Beispielprojekt „ac_1“) aus und basiert auf Annotationsspannen. Für die in dieser Collection mit „process“ annotierten Textpassagen gibt es in der zweiten *Annotation Collection* eine 100%-ige Überlappung im Sinne von Beispiel 3 für die letzte und vorletzte Textpassage. Für die erste Textspanne, die mit „process“ annotiert wurde, gibt es in der zweiten *Annotation Collection* ebenfalls eine 100%-ige Überlappung, nur dass die Passage hier mit „stative_event“ annotiert wurde.

Externe und weiterführende Links

- Anaconda: https://web.archive.org/save/https://www.anaconda.com/products/individual (Letzter Zugriff: 03.07.2024)
- CATMA GitLab: https://web.archive.org/save/https://git.catma.de/ (Letzter Zugriff: 03.07.2024)
- forTEXT.net GitHub-Repository: https://web.archive.org/save/hhttps://github.com/forTEXT/forTEXT.net (Letzter Zugriff: 03.07.2024)
- GitMA GitHub-Repository: https://web.archive.org/save/https://github.com/forTEXT/gitma (Letzter Zugriff: 03.07.2024)
- Pandas Dokumentation: https://web.archive.org/save/https://pandas.pydata.org/docs/index.html (Letzter Zugriff: 03.07.2024)

Bibliographie

Anaconda Software Distribution. 2020. Anaconda Documentation. https://docs.anaconda.com.

Artstein, Ron und Massimo Poesio. 2008. Inter-Coder Agreement for Computational Linguistics. *Computational Linguistics* 34, Nr. 4. doi: https://www.mitpressjournals.org/doi/pdfplus/10.1162/coli.07-034-R2,.

Bird, Steven, Ewan Klein und Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media.

Chacon, Scott und Ben Straub. 2014. *Pro Git*. Apress.

Gius, Evelyn, Jan Christoph Meister, Malte Meister, Marco Petris, Christian Bruck, Janina Jacke, Mareike Schumacher, Marie Flüh und Jan Horstmann. 2020. CATMA. 11. November. https://zenodo.org/records/4353618 (zugegriffen: 29. April 2022).

Horstmann, Jan. 2024. Lerneinheit: Manuelle Annotation mit CATMA. Hg. von Evelyn Gius. *forTEXT* 1, Nr. 4. Manuelle Annotation (7. August). doi: 10.48694/fortext.3750, https://fortext.net/routinen/lerneinheiten/manuelle-annotation-mit-catma.

Horstmann, Jan und Jan-Erik Stange. 2024. Methodenbeitrag: Textvisualisierung. Hg. von Evelyn Gius. *forTEXT* 1, Nr. 5. Textvisualisierung (7. August). doi: 10.48694/fortext.3772, https://fortext.net/routinen/methoden/textvisualisierung.

Jacke, Janina. 2024a. Methodenbeitrag: Kollaboratives literaturwissenschaftliches Annotieren. Hg. von Evelyn Gius. *forTEXT* 1, Nr. 4. Manuelle Annotation (7. August). doi: 10.48694/fortext.3749, https://fortext.net/routinen/methoden/kollaboratives-literaturwissenschaftliches-annotieren.

———. 2024b. Methodenbeitrag: Manuelle Annotation. Hg. von Evelyn Gius. *forTEXT* 1, Nr. 4. Manuelle Annotation (7. August). doi: 10.48694/fortext.3748, https://fortext.net/routinen/methoden/manuelle-annotation.

Kluyver, Thomas, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, u. a. 2016. Jupyter Notebooks – a publishing format for reproducible computational workflows. In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, hg. von F. Loizides und B. Schmidt, 87–90. IOS Press.

Mathet, Yann, Antoine Widlöcher und Jean-Philippe Métivier. 2015. The Unified and Holistic Method Gamma (γ) for Inter-Annotator Agreement Measure and Alignment. *Computational Linguistics* 41, Nr. 3 (1. September): 437–479. doi: 10.1162/COLI_a_00227, https://doi.org/10.1162/COLI_a_00227 (zugegriffen: 11. Mai 2023).

McKinney, Wes u. a. 2010. Data structures for statistical computing in python. In: *Proceedings of the 9th Python in Science Conference*, 445:51–56. Austin, TX.

Pagel, Janis, Nils Reiter, Ina Rösiger und Sarah Schulz. 2020. Annotation als flexibel einsetzbare Methode. In: *Interdisziplinäre(s) arbeiten in der CRETA-Werkstatt*, hg. von Nils Reiter, Axel Pichler, und Jonas Kuhn, 125–142. Berlin, Boston: De Gruyter. https://doi.org/10.1515/9783110693973-006 (zugegriffen: 19. Oktober 2023).

Schumacher, Mareike. 2024a. Toolbeitrag: CATMA. Hg. von Evelyn Gius. *forTEXT* 1, Nr. 4. Manuelle Annotation (7. August). doi: 10.48694/fortext.3761, https://fortext.net/tools/tools/catma.

———. 2024b. Methodenbeitrag: Netzwerkanalyse. Hg. von Evelyn Gius. *forTEXT* 1, Nr. 6. Netzwerkanalyse (30. August). doi: 10.48694/fortext.3759, https://fortext.net/routinen/methoden/netzwerkanalyse.

Titeux, Hadrien und Rachid Riad. 2021. pygamma-agreement: Gamma γ measure for inter/intra-annotator agreement in Python. *Journal of Open Source Software* 6, Nr. 62: 2989. doi: 10.21105/joss.02989, https://doi.org/10.21105/joss.02989.

Vauth, Michael, Malte Meister, Hans Ole Hatzel, Dominik Gerstorfer und Evelyn Gius. 2022. GitMA. Zenodo, 5. März. doi: 10.5281/ZENODO.6330464, https://doi.org/10.5281/ZENODO.6330464 (zugegriffen: 20. April 2022).

Glossar

Annotation Annotation beschreibt die manuelle oder automatische Hinzufügung von Zusatzinformationen zu einem Text. Die manuelle Annotation wird händisch durchgeführt, während die (teil-)automatisierte Annotation durch Machine-Learning-Verfahren durchgeführt wird. Ein klassisches Beispiel ist das automatisierte PoS-Tagging (Part-of-Speech-Tagging), welches oftmals als Grundlage (Preprocessing) für weitere Analysen wie Named Entity Recognition (NER) nötig ist. Annotationen können zudem deskriptiv oder analytisch sein.

API API steht für *Application Programming Interface* und bezeichnet eine Programmierschnittstelle, die Soft- und Hardwarekomponenten wie Anwendungen, Festplatten oder Benutzeroberflächen verbindet. Sie vereinheitlicht die Datenübergabe zwischen Programmteilen, etwa Modulen, und Programmen.

Browser Mit Browser ist in der Regel ein Webbrowser gemeint, also ein Computerprogramm, mit dem das Anschauen, Navigieren auf, und Interagieren mit Webseiten möglich wird. Am häufigsten genutzt werden dafür Chrome, Firefox, Safari oder der Internet Explorer.

Commandline Die Commandline (engl. *command line interface* (CLI)), auch Kommandozeile, Konsole, Terminal oder Eingabeaufforderung genannt, ist die direkteste Methode zur Interaktion eines Menschen mit einem Computer. Programme ohne eine grafische Benutzeroberfläche (GUI) werden i. d. R. durch Texteingabe in

die Commandline gesteuert. Um die Commandline zu öffnen, klicken Sie auf Ihrem Mac „cmd“ + „space“, geben „Terminal“ ein und doppelklicken auf das Suchergebnis. Bei Windows klicken Sie die Windowstaste + „R“, geben „cmd.exe“ ein und klicken Enter.

CSV CSV ist die englische Abkürzung für *Comma Separated Values*. Es handelt sich um ein Dateiformat zur einheitlichen Darstellung und Speicherung von einfach strukturierten Daten mit dem Kürzel `.csv` , sodass diese problemlos zwischen IT-Systemen ausgetauscht werden können. Dabei sind alle Daten zeilenweise angeordnet. Alle Zeilen wiederum sind in einzelne Datenfelder aufgeteilt, welche durch Trennzeichen wie Semikola oder Kommata getrennt werden können. In Programmen wie Excel können solche Textdateien als Tabelle angezeigt werden.

Feature Unter Features können Einzelfunktionen eines Tools verstanden werden, die beispielsweise komplexe Funktionen wie die Visualisierung eines Textes als Wordcloud ermöglichen, oder auch kleinere Funktionseinheiten wie den Abgleich einzelner Spracheigenschaften (Properties) mit annotierten Beispieltexten darstellen.

GUI GUI steht für *Graphical User Interface* und bezeichnet eine grafische Benutzeroberfläche. Ein GUI ermöglicht es, Tools mithilfe von grafischen Schaltflächen zu bedienen, um somit beispielsweise den Umgang mit der Commandline zu umgehen.

JSON JSON ist die englische Abkürzung für *JavaScript Object Notation* . Dabei handelt es sich um ein kompaktes Textformat, das insbesondere zum Datenaustausch entworfen wurde. Es ist für Menschen einfach zu lesen und zu schreiben und für Maschinen einfach zu analysieren und zu generieren. JSON ist ein Format, das unabhängig von Programmiersprachen ist.

Lemmatisieren Die Lemmatisierung von Textdaten gehört zu den wichtigen Preprocessing-Schritten in der Textverarbeitung. Dabei werden alle Wörter (Token) eines Textes auf ihre Grundform zurückgeführt. So werden beispielsweise Flexionsformen wie „schneller“ und „schnelle“ dem Lemma „schnell“ zugeordnet.

Machine Learning Machine Learning, bzw. maschinelles Lernen im Deutschen, ist ein Teilbereich der künstlichen Intelligenz. Auf Grundlage möglichst vieler (Text-)Daten erkennt und erlernt ein Computer die häufig sehr komplexen Muster und Gesetzmäßigkeiten bestimmter Phänomene. Daraufhin können die aus den Daten gewonnen Erkenntnisse verallgemeinert werden und für neue Problemlösungen oder für die Analyse von bisher unbekannten Daten verwendet werden.

Named Entities Eine Named Entity (NE) ist eine Entität, oft ein Eigenname, die meist in Form einer Nominalphrase zu identifizieren ist. Named Entities können beispielsweise Personen wie „Nils Holgerson“, Organisationen wie „WHO“ oder Orte wie „New York“ sein. Named Entities können durch das Verfahren der Named Entity Recognition (NER) automatisiert ermittelt werden.

POS PoS steht für *Part of Speech* , oder „Wortart“ auf Deutsch. Das PoS- Tagging beschreibt die (automatische) Erfassung und Kennzeichnung von Wortarten in einem Text und ist of ein wichtiger Preprocessing-Schritt, beispielsweise für die Analyse von Named Entities.

Preprocessing Für viele digitale Methoden müssen die zu analysierenden Texte vorab „bereinigt“ oder „vorbereitet“ werden. Für statistische Zwecke werden Texte bspw. häufig in gleich große Segmente unterteilt (*chunking*), Großbuchstaben werden in Kleinbuchstaben verwandelt oder Wörter werden lemmatisiert.

Programmiercode Der Code, oder auch Programmcode/ Maschinencode, bezieht sich auf eine Sammlung von Anweisungen, die durch verschiedene Programmiersprachen wie Java, Python oder C realisiert werden können. Für die Ausführung der Anweisungen wird der Code durch einen Compiler oder einen Interpreter in die Maschinensprache, einen Binärcode, des Computers übersetzt.

Property Property steht für „Eigenschaft“, „Komponente“ oder „Attribut“. In der automatischen Annotation dienen konkrete Worteigenschaften wie Groß- und Kleinschreibung zur Klassifizierung von Wörtern oder Phrasen. Durch die Berücksichtigung solcher Eigenschaften in den Features eines Tools kann maschinelles Lernen bestimmter Phänomene umgesetzt werden. In der manuellen Annotation können als Properties auch Eigenschaften von Annotationen benannt werden.

Stoppwortliste Stoppwörter sind hochfrequente Wörter, meist Funktionswörter, die, aufgrund ihrer grammatisch bedingten Häufigkeit, beispielsweise die Ergebnisse von inhaltlichen oder thematischen Analysen verzerren können. Deshalb werden diese Wörter, gesammelt in einer Stoppwortliste, bei digitalen Textanalysen meist nicht berücksichtigt.

Tagset Ein Tagset definiert die Taxonomie, anhand derer Annotationen in einem Projekt erstellt werden. Ein Tagset beinhaltet immer mehrere Tags und ggf. auch Subtags. Ähnlich der Type/Token -Differenz in der Linguistik sind Tags deskriptive Kategorien, wohingegen Annotationen die einzelnen Vorkommnisse dieser Kategorien im Text sind.

Type/Token Das Begriffspaar „Type/Token“ wird grundsätzlich zur Unterscheidung von einzelnen Vorkommnissen (Token) und Typen (Types) von Wörtern oder Äußerungen in Texten genutzt. Ein Token ist also ein konkretes Exemplar eines bestimmten Typs, während ein Typ eine im Prinzip unbegrenzte Menge von

Exemplaren (Token) umfasst.
Es gibt allerdings etwas divergierende Definitionen zur Type-Token-Unterscheidung. Eine präzise Definition ist daher immer erstrebenswert. Der Satz „Ein Bär ist ein Bär.“ beinhaltet beispielsweise fünf Worttoken („Ein“, „Bär“, „ist“, „ein“, „Bär“) und drei Types, nämlich: „ein“, „Bär“, „ist“. Allerdings könnten auch vier Types, „Ein“, „ein“, „Bär“ und „ist“, als solche identifiziert werden, wenn Großbuchstaben beachtet werden.

Wordcloud Eine *Wordcloud* , oder auch Schlagwortwolke, ist eine Form der Informationsvisualisierung, beispielsweise von Worthäufigkeiten in einem Text oder einer Textsammlung. Dabei werden unterschiedlich gewichtete Wörter, wie die häufigsten Wörter, i.d.R. größer oder auf andere Weise hervorgehoben dargestellt. Die horizontale/vertikale Ausrichtung und die Farbe der dargestellten Wörter hat meistens allerdings keinen semantischen Mehrwert.

ZIP ZIP steht für ein Dateiformat (zip = engl. Reißverschluss), in welchem mehrere Einzeldateien verlustfrei, komprimiert zusammengefasst werden. ZIP-Dateien werden beim Öffnen entweder automatisch entpackt oder lassen sich per Rechtsklick extrahieren.