Deep Learning

Übung WS 23/24

Judith Nester (nester@uni-koeln.de)

19-10-2023

Section 1

Hello World

Judith Nester

- Wissenschaftliche Mitarbeiterin am IDH und am RRZK
- Promotion im Bereich Discriminatory Speech Detection in politischer Sprache
- Forschungsinteressen
  - Deep Learning
  - Natural Language Generation und Natural Language Processing
  - Korpuslinguistik
  - Datenannotation

Section 2

Formalitäten und Ablauf

Computerlinguistik im B.A. Informationsverarbeitung

- Modul **Grundlagen der Computerlinguistik** (alte Studienordnung "Computerlinguistische Grundlagen")
- Modul **Anwendungen der Computerlinguistik** (alte Studienordnung "Angewandte Linguistische Datenverarbeitung")

Computerlinguistik im B.A. Informationsverarbeitung

- Modul **Grundlagen der Computerlinguistik** (alte Studienordnung "Computerlinguistische Grundlagen")
  - **Seminar** Computerlinguistische Grundlagen (immer im WiSe, Dozent Hermes, Inhalt: Linguistische Grundlagen, Annotation)
  - **Vorlesung** Sprachverarbeitung (immer im SoSe, Dozent Reiter, Quantitative Eigenschaften von Sprache, Machine Learning)
  - **Übung** Sprachverarbeitung (immer im SoSe, Dozent Reiter, begleitend zur Vorlesung, früher Seminar II)
  - **Modulprüfung** Klausur (immer im SoSe, 90 Minuten, Teilleistung im WiSe möglich, 30 Minuten)

Computerlinguistik im B.A. Informationsverarbeitung

- Modul **Anwendungen der Computerlinguistik** (alte Studienordnung "Angewandte Linguistische Datenverarbeitung")
  - **Übung** Deep Learning (immer im WiSe, Dozentin Nester, Inhalt: Deep Learning Methoden)
  - **Hauptseminar** Experimentelles Arbeiten in der Sprachverarbeitung (immer im WiSe, Dozent Reiter, Inhalt: Experimente in der CL, wo kommen Fortschritt und Erkenntnis her?)
  - **Modulprüfung** Programmierhausarbeit

Kursablauf

Lernziele

- Code mit git versionieren
- Python-Code lesen und schreiben (oberflächlich; das ist kein Python-Kurs!)
- Erste Deep-Learning-Experimente mit tensorflow durchführen
- Verschiedene DL-Architekturen (auch konzeptuell) kennenlernen

Kursablauf

Lernziele

- Code mit git versionieren
- Python-Code lesen und schreiben (oberflächlich; das ist kein Python-Kurs!)
- Erste Deep-Learning-Experimente mit tensorflow durchführen
- Verschiedene DL-Architekturen (auch konzeptuell) kennenlernen

Studienleistung

- Aufgaben abgeben
- Aktive Teilnahme, experimentieren, sich trauen auch mal was Unfertiges abzugeben
- Keine Modulprüfung in diesem Kurs!

Kursablauf

- In jeder Sitzung:
  - Einführung in neues Thema
  - Vorstellung der Übungsaufgabe
  - Lab session: Arbeit in Kleingruppen an der Übung
  - Fertigstellen der Übung bis zur angegebenen Frist
    - Abgabe der Übungen via GitHub
  - Kommentierte Referenzlösung erscheint später als update via GitHub

Kursorganisation

Ressourcen, Literatur, Kommunikation

- Kurswebseite: `https://lehre.idh.uni-koeln.de/lehrveranstaltungen/wintersemester-2023-2024/deep-learning/`
  - Folien, Zeitplan, Literatur
- GitHub-Repository: `https://github.com/IDH-Cologne-Deep-Learning-Uebung/uebung-deep-learning-ws2324`
  - Übungen erscheinen dort als Repository
  - Bitte User anlegen und in die Tabelle eintragen: `https://docs.google.com/spreadsheets/d/1WGK87XckHrmd2115IP7SgNHbBDykMQAZYHp1Du0yThM/edit?usp=sharing`

Kursorganisation

Ressourcen, Literatur, Kommunikation

- Kurswebseite: `https://lehre.idh.uni-koeln.de/lehrveranstaltungen/wintersemester-2023-2024/deep-learning/`
  - Folien, Zeitplan, Literatur
- GitHub-Repository: `https://github.com/IDH-Cologne-Deep-Learning-Uebung/uebung-deep-learning-ws2324`
  - Übungen erscheinen dort als Repository
  - Bitte User anlegen und in die Tabelle eintragen: `https://docs.google.com/spreadsheets/d/1WGK87XckHrmd2115IP7SgNHbBDykMQAZYHp1Du0yThM/edit?usp=sharing`
- E-Mail: nester@uni-koeln.de
  - Alles andere

Section 3

Introduction

Geplante Themen

- Version Control (Funktionsweise, Workflow, git)

Geplante Themen

- Version Control (Funktionsweise, Workflow, git)
- Einführung in Python (Syntax, Datentypen, List Comprehension, Functions, Classes, Libs und Packages, Exceptions)

Geplante Themen

- Version Control (Funktionsweise, Workflow, git)
- Einführung in Python (Syntax, Datentypen, List Comprehension, Functions, Classes, Libs und Packages, Exceptions)
- Deep Learning
  - Tasks
  - Lineare und logistische Regression
  - Feed-forward neural networks
  - Recurrent neural networks
  - Overfitting, Dropout, Regularization
  - Input Representation, Embeddings
  - LSTMs
  - Encoder/Decoder networks
  - Attention
  - Transformer

Today's To Do

» Deep Learning - Worum geht es hier eigentlich?

Today's To Do

- Deep Learning - Worum geht es hier eigentlich?
- Git - In den Workflow kommen

Today's To Do

- Deep Learning - Worum geht es hier eigentlich?
- Git - In den Workflow kommen
- Kleine Aufgabe

Section 4

Deep Learning

Der Begriff Deep Learning

Trend, buzz word, vielgenanntes Thema in Wirtschaft und Politik

Der Begriff Deep Learning

Trend, buzz word, vielgenanntes Thema in Wirtschaft und Politik

- Was ist Deep Learning?
- Wie grenzt sich Deep Learning von Künstlicher Intelligenz und Machine Learning ab?

Künstliche Intelligenz

Künstliche Intelligenz

» Systeme und Programme, die menschliche Intelligenz imitieren

Künstliche Intelligenz

- Systeme und Programme, die menschliche Intelligenz imitieren
- Erwähnung zuerst 1955 (John McCarthy, Dartmouth Summer Research Project on Artificial Intelligence)

Künstliche Intelligenz

- Systeme und Programme, die menschliche Intelligenz imitieren
- Erwähnung zuerst 1955 (John McCarthy, Dartmouth Summer Research Project on Artificial Intelligence)
- Schwache vs. starke KI

Künstliche Intelligenz

- Systeme und Programme, die menschliche Intelligenz imitieren
- Erwähnung zuerst 1955 (John McCarthy, Dartmouth Summer Research Project on Artificial Intelligence)
- Schwache vs. starke KI
  - Schwach:
    - Konkrete Anwendungsprobleme (Schach, Go, etc.)
    - Nicht wirklich "intelligent"

Künstliche Intelligenz

- Systeme und Programme, die menschliche Intelligenz imitieren
- Erwähnung zuerst 1955 (John McCarthy, Dartmouth Summer Research Project on Artificial Intelligence)
- Schwache vs. starke KI
  - Schwach:
    - Konkrete Anwendungsprobleme (Schach, Go, etc.)
    - Nicht wirklich "intelligent"
  - Stark:
    - Auf Augenhöhe mit dem Menschen (oder intelligenter?)
    - Eigenständig, kreativ, emotional, sozial, sensomotorisch und kognitiv, etc.
    - SciFi!

Machine Learning

- Teilgebiet Künstlicher Intelligenz
- Systeme und Programme, die aus großen Datenmengen lernen und die gesammelte Erfahrung zur Lösung bestimmter Probleme einsetzen können
- Einsatz in Bilderkennung (z.B. Gesichtserkennung, MRT-Analyse), Malware-Detection, Spam-Filter, Wettervorhersagen, u.v.m.
- Einteilung in Supervised Learning, Unsupervised Learning und Reinforcement Learning

Machine Learning

Supervised Learning

Machine Learning

Supervised Learning

- Trainings- und Testdaten für Lernprozess
- Daten enthalten bereits die gewünschten Ergebnisse, sprich gelabelte Daten (bspw. Bilddaten mit Labels HUND oder KATZE)
- Funktion passt sich selbstständig an, so dass sie vom ungelabelten Input auf den gewünschten, bekannten Output kommt
- Verifizierung mit Testdaten
- Nachteil: Hoher Aufwand in der Datenaufbereitung
- Vorteil: Besser nachvollziehbar und evaluierbar

Machine Learning

Unsupervised Learning

Machine Learning

Unsupervised Learning

- Erlernen bestimmter Muster anhand unbestimmter Merkmale in Eingangsdaten –> Bildung von Vektor-Clustern
- Vollautomatisierte Erstellung von Modellen
- Nachteil: Schwere Nachvollziehbarkeit, darum aufwendige Tests nötig
- Vorteil: Keine aufwendige Datenaufbereitung

Machine Learning

Reinforcement Learning

Machine Learning

Reinforcement Learning

- Belohnungsprinzip
- Selbstständiges Erlernen einer Strategie, um "Belohnungen" zu erhalten
- Funktion, die bestimmte Zustände im Verlauf einer Aktion positiv oder negativ bewerten kann
- Einsatz bspw. bei Backgammon-Automat

Deep Learning

» Teilgebiet von Machine Learning

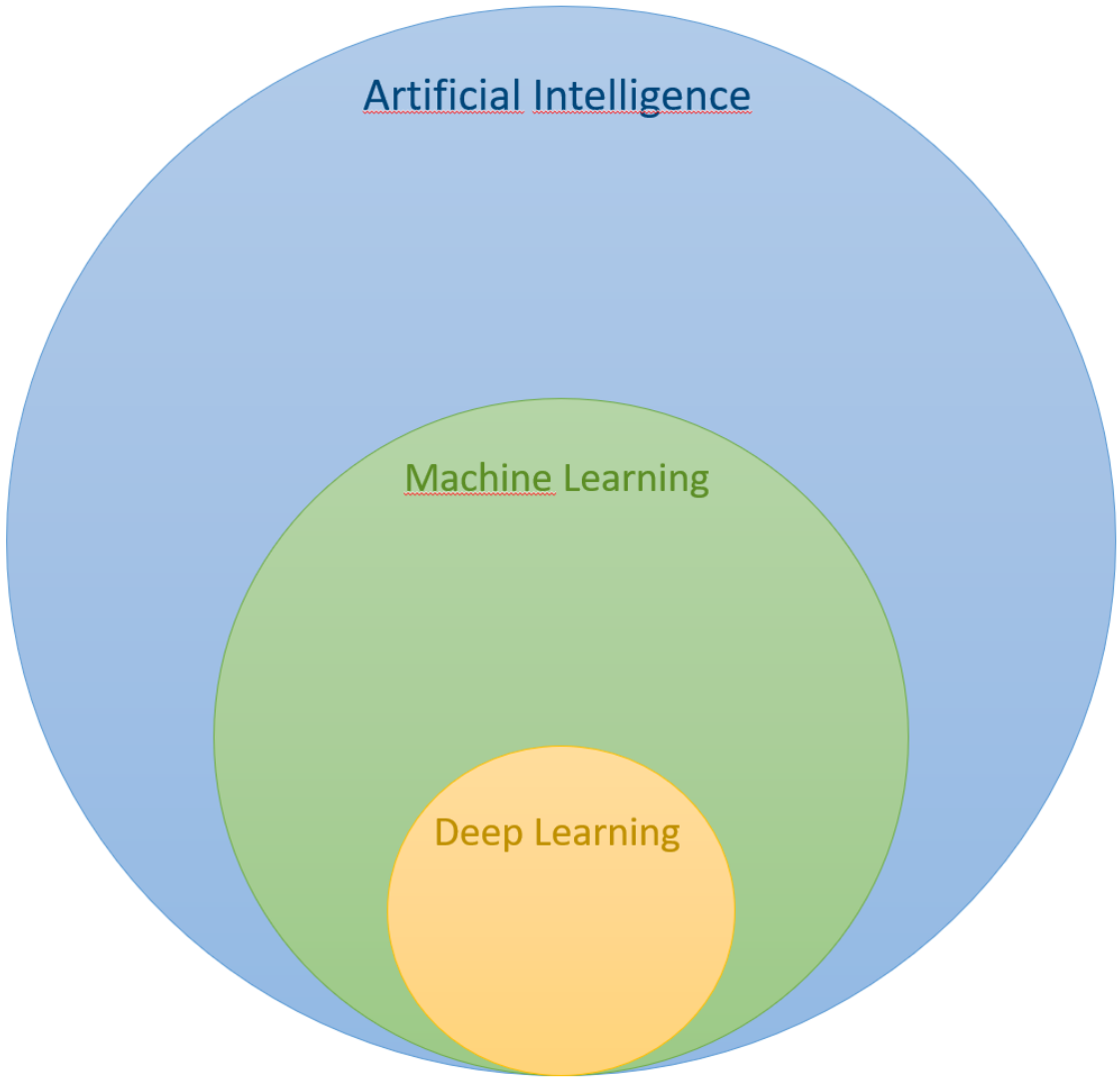
Deep Learning

- Teilgebiet von Machine Learning
- Umsetzung maschinellen Lernens mit Hilfe großer Datenmengen und künstlichen neuronalen Netzen
  - Neuronale Netze als Repräsentation des Lernprozesses
  - Input geht in ein Neuron ein -> Anwendung mathematischer Funktionen -> Output geht als Input ins nächste Neuron ein

Deep Learning

- Teilgebiet von Machine Learning
- Umsetzung maschinellen Lernens mit Hilfe großer Datenmengen und künstlichen neuronalen Netzen
  - Neuronale Netze als Repräsentation des Lernprozesses
  - Input geht in ein Neuron ein -> Anwendung mathematischer Funktionen -> Output geht als Input ins nächste Neuron ein
- Input-Daten werden in Form von Vektoren in einem hochdimensionalen Vektorraum abgebildet (Embeddings)
- Benötigt sehr große Datenmengen für Lernprozesse -> representation learning

Deep Learning



Section 5

Version Control

Version control

- Versioning of source code
- Differences between versions
- Maintaining several branches in parallel

Version control

- Versioning of source code
- Differences between versions
- Maintaining several branches in parallel

Why is this useful?

- Programming projects quickly become massive
  - Windows 2000: 28mio LoC (ca. 930k standard pages)
  - CorefAnnotator: 27k LoC (ca. 770 standard pages)

Version control

- Versioning of source code
- Differences between versions
- Maintaining several branches in parallel

Why is this useful?

- Programming projects quickly become massive
  - Windows 2000: 28mio LoC (ca. 930k standard pages)
  - CorefAnnotator: 27k LoC (ca. 770 standard pages)
- Large teams
  - working on the same project
  - over a long time (don't rely on human memory)

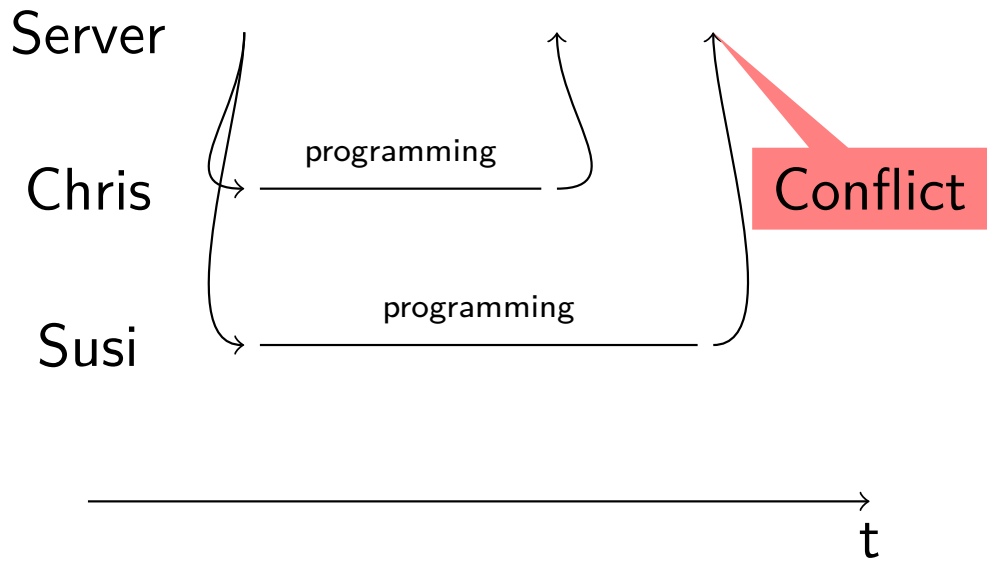
Version control

- Versioning of source code
- Differences between versions
- Maintaining several branches in parallel

Why is this useful?

- Programming projects quickly become massive
  - Windows 2000: 28mio LoC (ca. 930k standard pages)
  - CorefAnnotator: 27k LoC (ca. 770 standard pages)
- Large teams
  - working on the same project
  - over a long time (don't rely on human memory)
- A single conceptual change often distributed over many files

Situations



Conflict resolution options

- Ignore, let Susi overwrite Chris' code (this is bad!)
- Create a second copy (this is what Dropbox does)
- Force Susi to *explicitly* merge the code

What do we put under version control?

plain text files

- source code (python, java, perl, c, . . .)
- texts (plain, latex, markdown)
- primary data (xml, csv)
  - but beware of large files
- vector graphics (svg)

What do we put under version control?

plain text files

- source code (python, java, perl, c, . . .)
- texts (plain, latex, markdown)
- primary data (xml, csv)
  - but beware of large files
- vector graphics (svg)

Don't put these in VC:

Binary files

- word documents, pdf files
- images (jpg, png)
- compiled code (executables)

Software

- Very old
  - CVS (concurrent versioning system)
  - Rarely used today
- Old
  - SVN (subversion)
  - Sometimes used
- State of the art
  - git
- More solutions are available commercially

git

- Developed by the Linux kernel developers
- Open source – `https://git-scm.com`
- Distributed
  - Each user has their own local version of the entire repository on their computer
  - No central server required
    - . . . but still useful to have one
- Fast
- Data assurance
  - Checksums to identify revisions (commits) and to make sure you get out what you put in
  - Manipulations in the revision history mean a change in the checksum and thus a traceability of the change

git vs. GitHub vs. GitLab

- git is an open source software
  - `https://git-scm.com`
- GitHub is a (commercial) web platform
  - Recently bought by Microsoft
  - GitHub provides a central server for git repositories *and* additional services (wiki, ticket system, ...)
  - `https://github.com`
- GitLab is an open source software
  - Provides a central server that you can install on your own server (e.g., at the IDH)
  - `https://about.gitlab.com`

Reading

Scott Chacon and Ben Straub: "Pro Git". 2nd edition. Apress, 2014.
`https://git-scm.com/book/en/v2`

Table of Contents

1. Getting Started
2. Git Basics
3. Git Branching
4. Git on the Server
5. Distributed Git
6. ...

Subsection 1

How does git work?

Commit

- One version of an entire directory (including subdirectories)
- Creating commits is the central activity we do
- Each commit knows its predecessor
- Each commit is identified by a hash value:
  `0eabb4bfef80be2af18255dc19301b989da1f1a3`
- A commit can include changes in multiple files

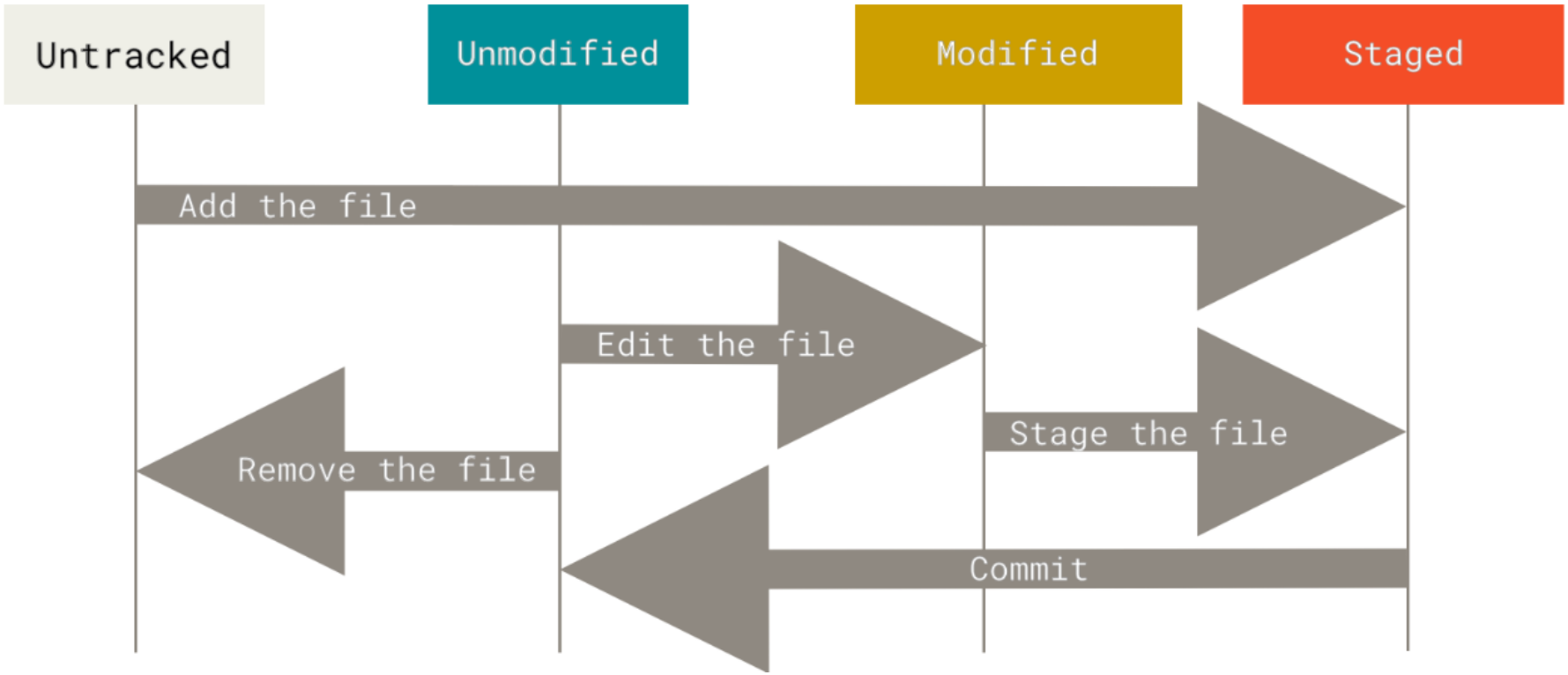


Figure: The lifecycle of the status of your files (Chacon/Straub: Pro Git)

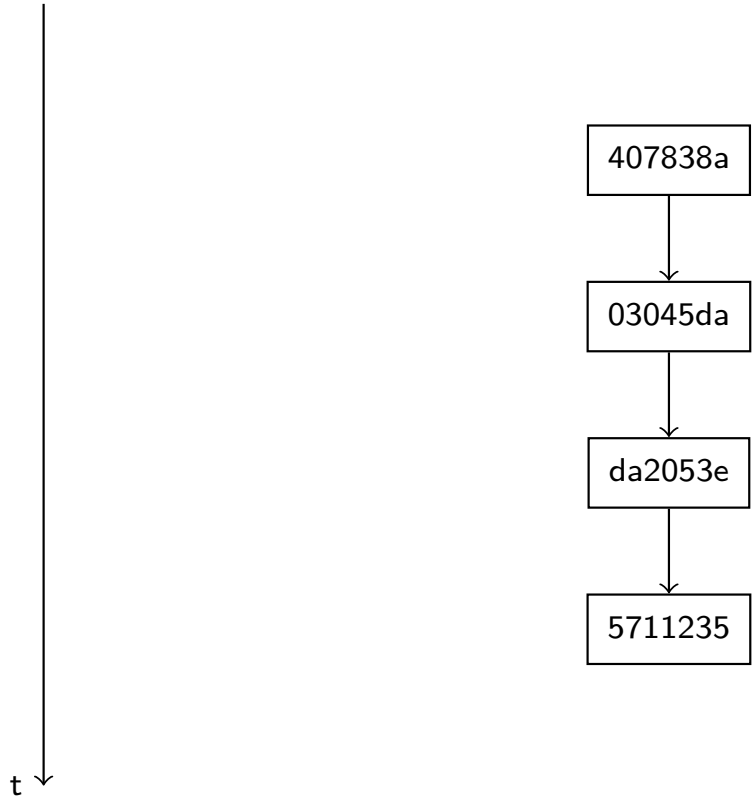
Workflow

1. (Pull changes from others)
2. Edit/add files
3. Put files in staging area
   - `git add <FILENAME>`
   - `git remove <FILENAME>`
4. Commit all files in staging area
   - Provide a useful description
   - `git commit -m "comment"`
5. (Push to others)

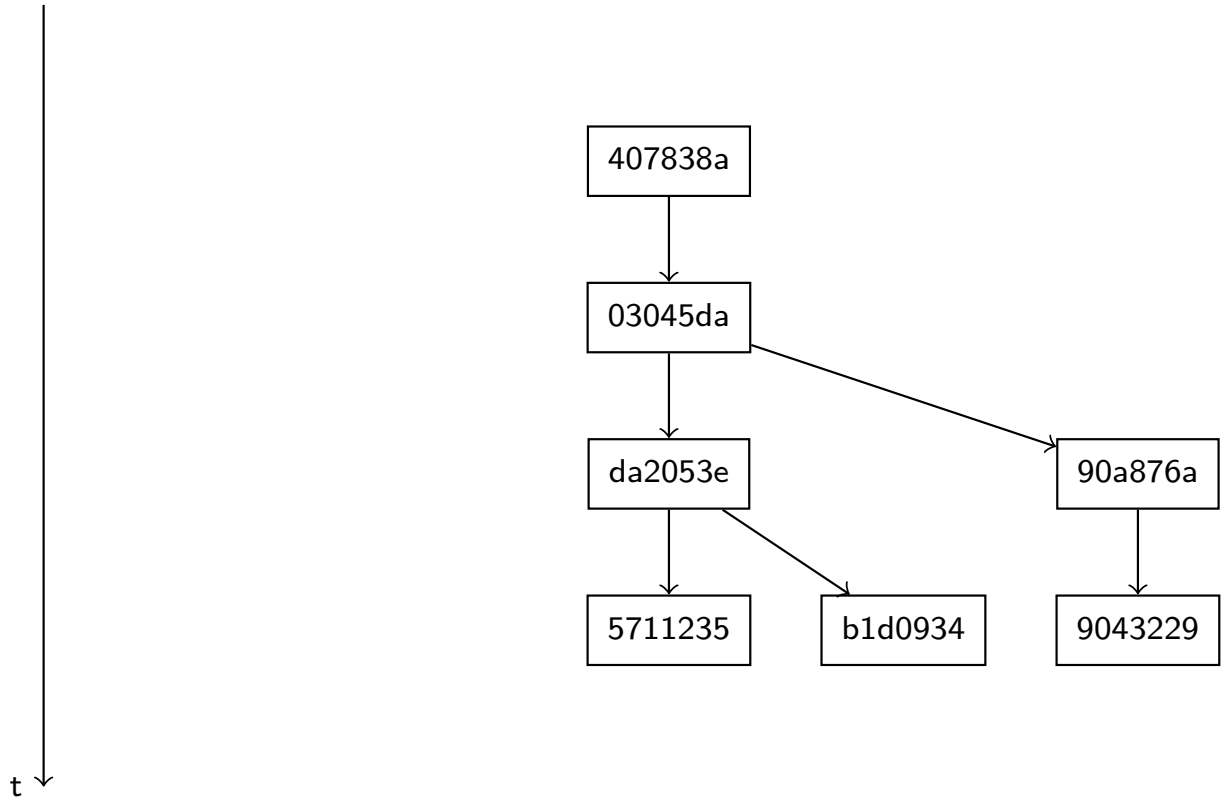
Branching

- Maintaining multiple branches is often useful
- At each time, a single branch is active
  - By default: `master` or `main`
- Switch to an existing branch
  - `git checkout <BRANCHNAME>`
  - To create a new branch, add the option `-b`:
    - `git checkout -b <BRANCHNAME>`

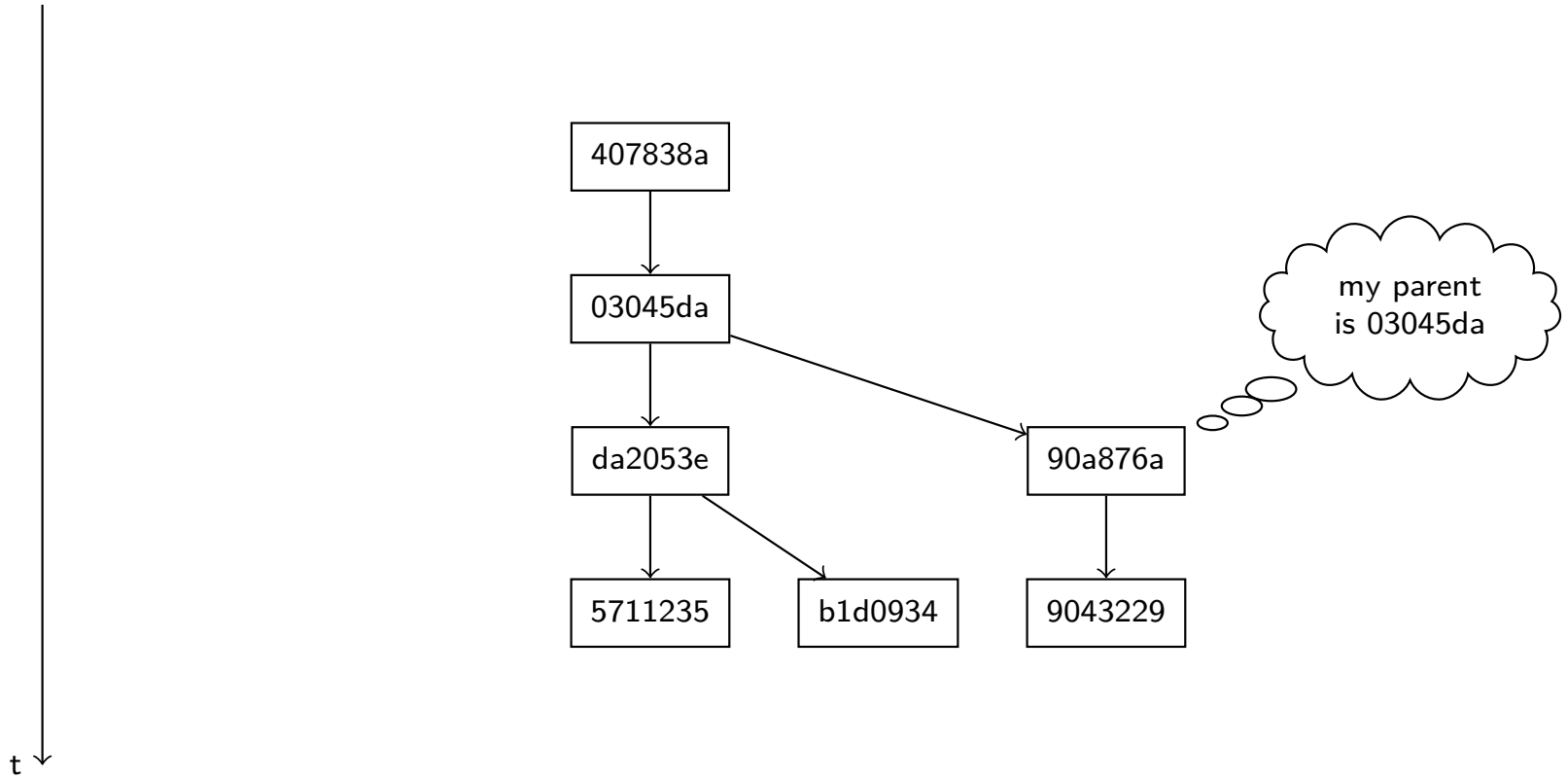
Branching and committing results in a tree



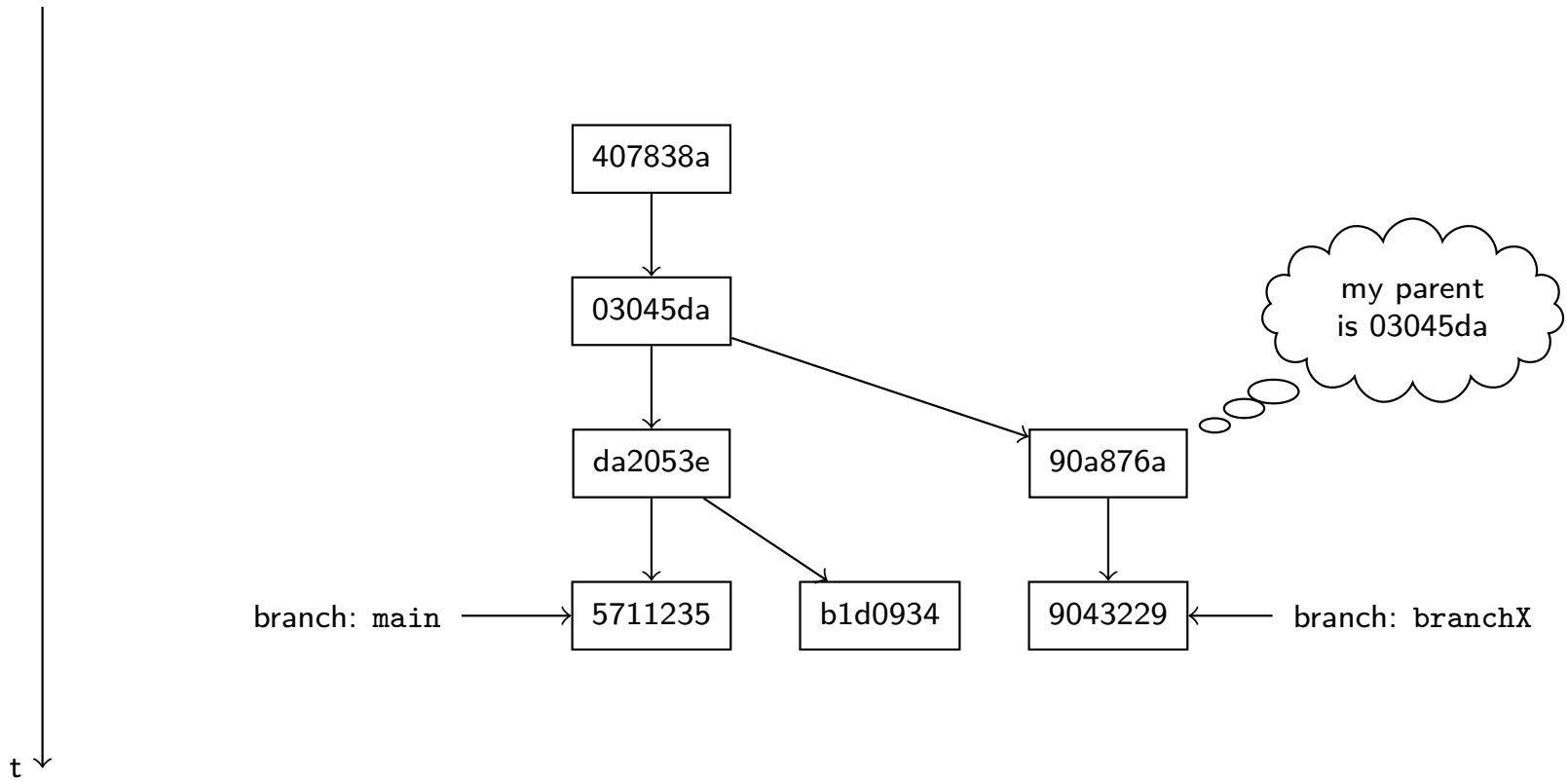
Branching and committing results in a tree



Branching and committing results in a tree



Branching and committing results in a tree



Repository vs. working copy

- The git repository keeps track of *all* past versions and branches
- The working copy can be set to any of the past versions
- `git checkout REFNAME`
  - `REFNAME` can be a branch or revision hash (or tag)
- Checking out moves the HEAD pointer to another revision
  - The HEAD pointer always points to the revision that's active in your working copy

Remotes

- Git repositories can be associated with *remote* repositories
  - Remote repositories are usually on a different computer (e.g., GitHub)

Remotes

- Git repositories can be associated with *remote* repositories
  - Remote repositories are usually on a different computer (e.g., GitHub)
- A repository needs to be synchronized with its remote manually:
  - `git push`: Transfers the commits on the local branch to the same branch on the remote
  - `git pull`: Transfers the commits on the remote branch to the local branch
  - `git clone REPOURL`: Create a local copy of the repository url, setting REPOURL as 'origin' remote

Useful commands

`git status`

Shows the status of the current working copy

- Changed files
- Files in the staging area
- The current branch

Useful commands

`git status`

Shows the status of the current working copy

- Changed files
- Files in the staging area
- The current branch

`git log`

Shows information about current and past commits
Useful options:

`-oneline` Each commit is shown on a single line

`-graph` Information is rendered visually

`-all` Shows information about all branches

On GUIs

Git has a complex task and is a complex piece of software

- Graphical user interfaces do exist and make some tasks easier
- In this class: command line

On GUIs

Git has a complex task and is a complex piece of software

- Graphical user interfaces do exist and make some tasks easier
- In this class: command line
- Recommendations
  - SourceTree (Win/Mac): `https://www.sourcetreeapp.com`
    - Needs a registration with BitBucket (similar to GitHub), but free
  - GitKraken (Win/Mac/Lin): `https://www.gitkraken.com`
    - Free for open source projects

On GUIs

Git has a complex task and is a complex piece of software

- Graphical user interfaces do exist and make some tasks easier
- In this class: command line
- Recommendations
  - SourceTree (Win/Mac): `https://www.sourcetreeapp.com`
    - Needs a registration with BitBucket (similar to GitHub), but free
  - GitKraken (Win/Mac/Lin): `https://www.gitkraken.com`
    - Free for open source projects
- More can be found here: `https://git-scm.com/downloads/guis/`

Exercise 01

- Readme im Repository unter exercise-01:
  `https://github.com/IDH-Cologne-Deep-Learning-Uebung/exercise-01`