Docker

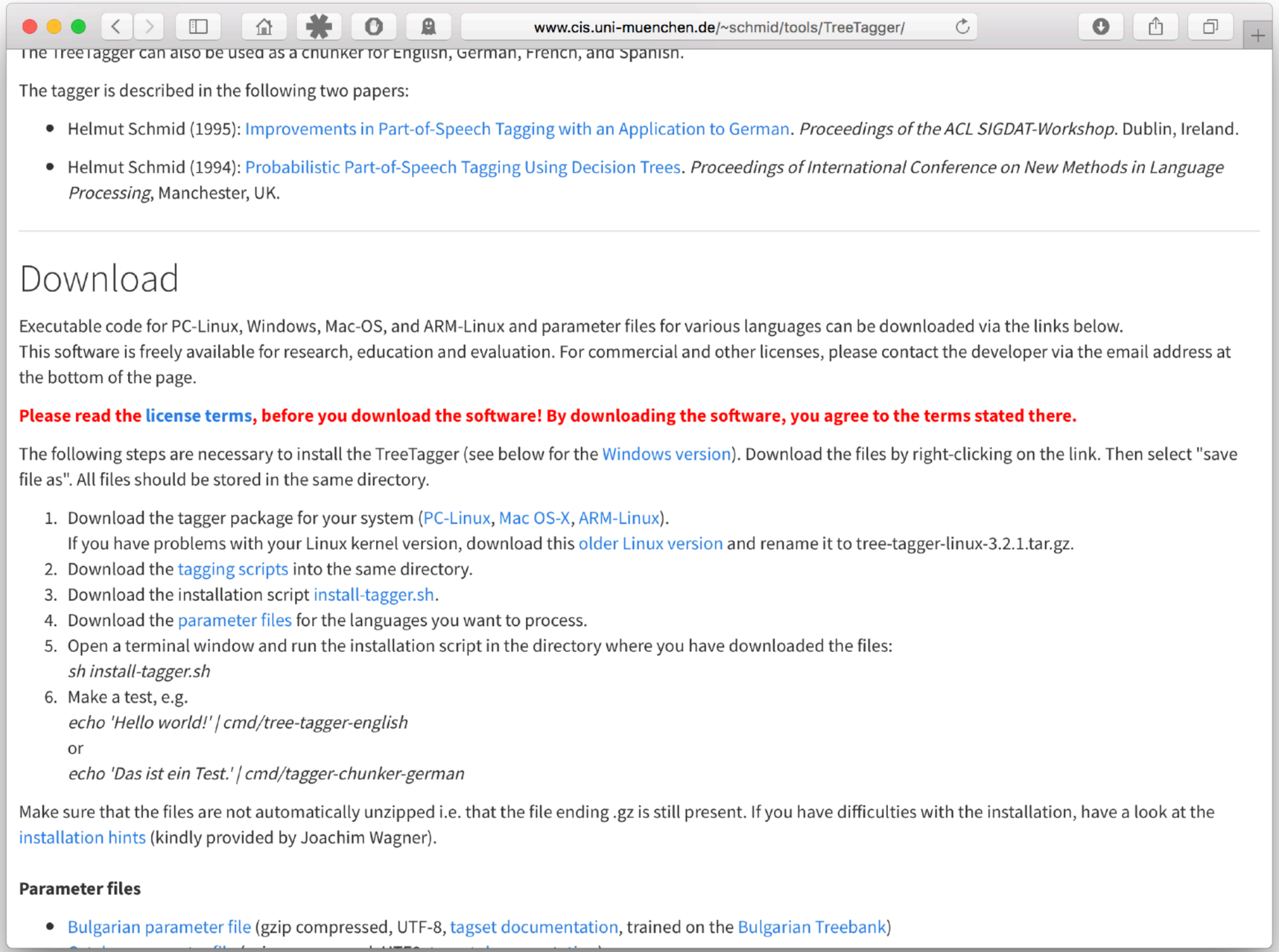
Proseminar “Tools”

Wintersemester 2017/18

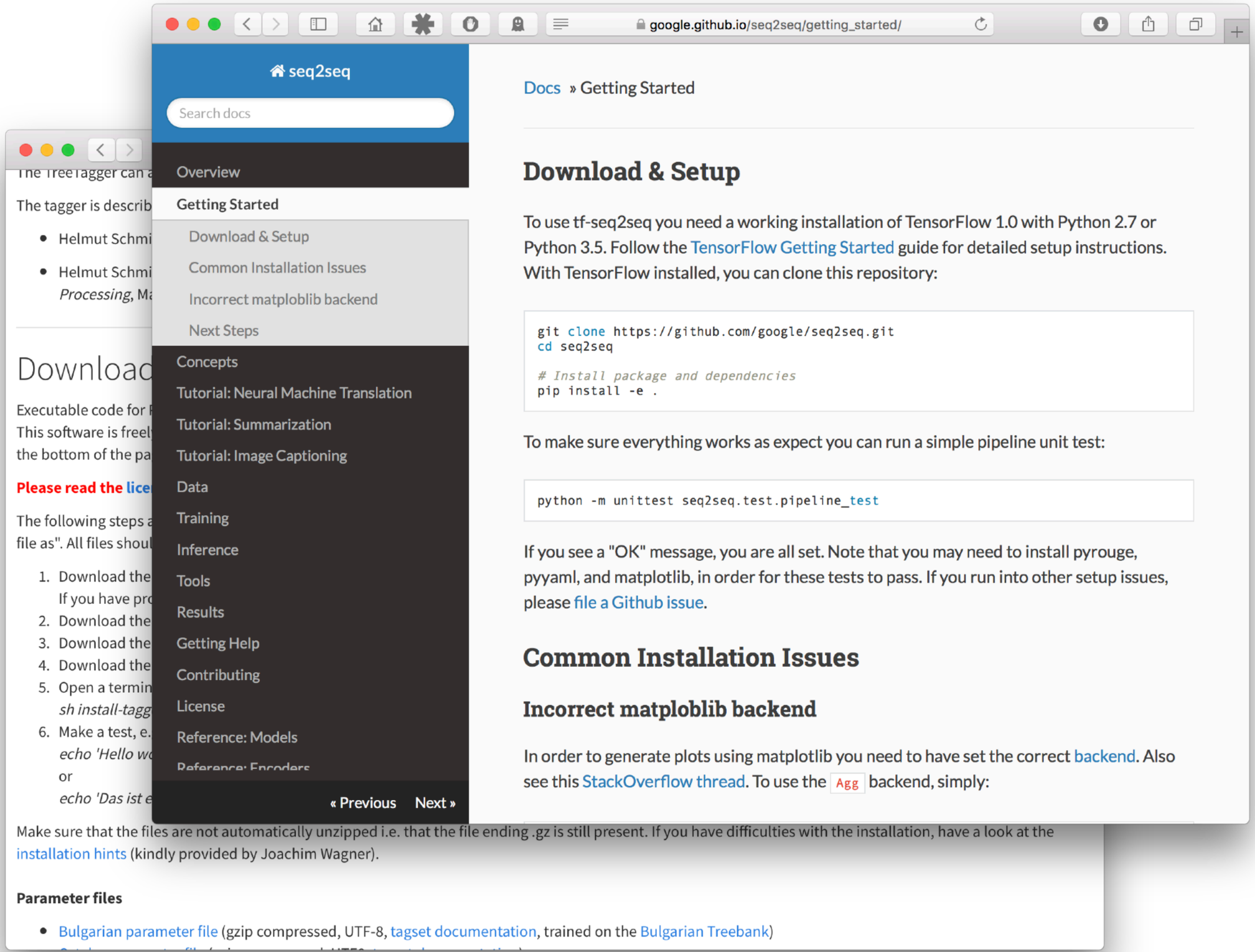
Alexander Koller

3. November 2017

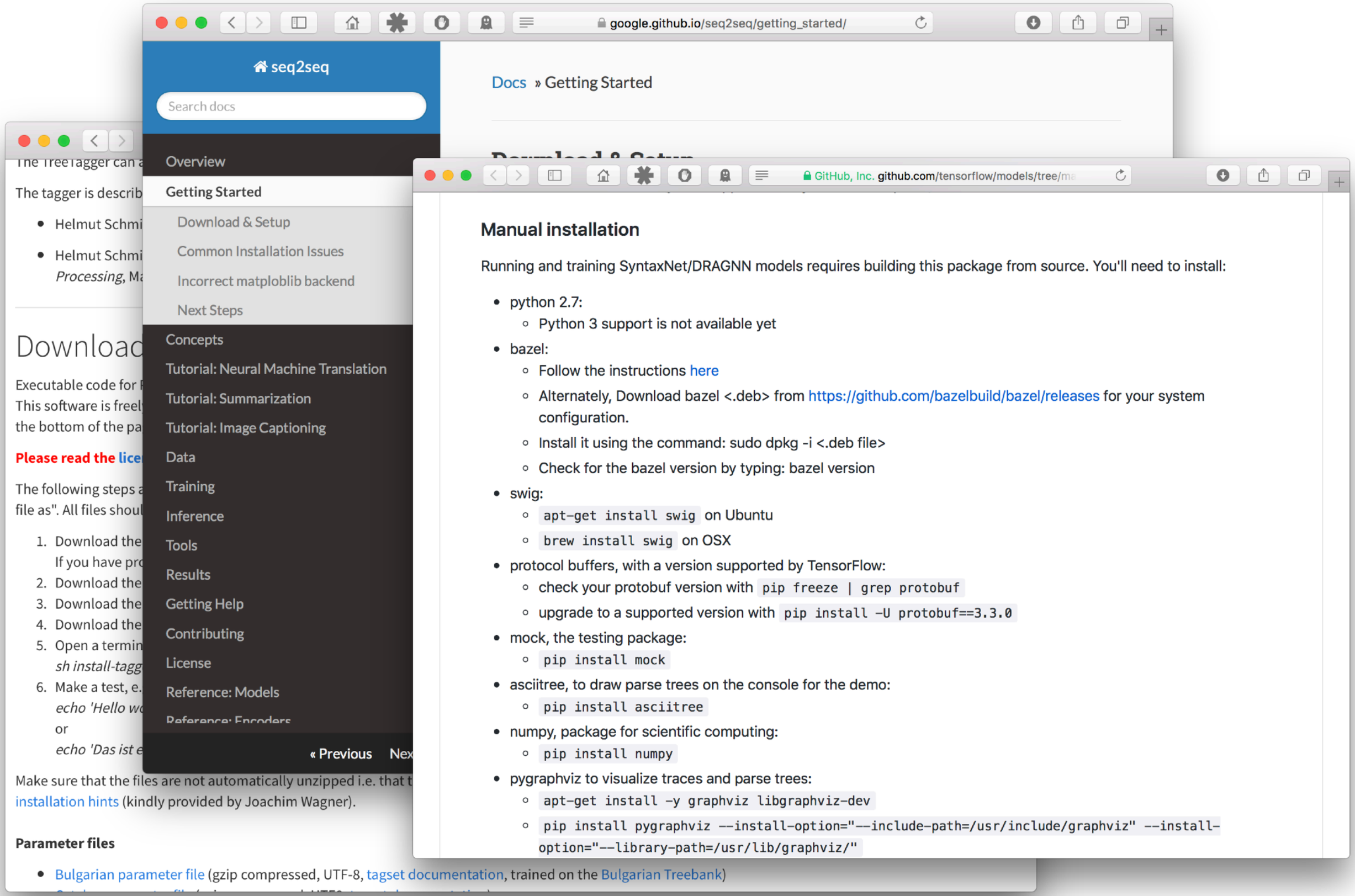
Das Problem


www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/
The TreeTagger can also be used as a chunker for English, German, French, and Spanish.
The tagger is described in the following two papers:
Helmut Schmid (1995): Improvements in Part-of-Speech Tagging with an Application to German. Proceedings of the ACL SIGDAT-Workshop. Dublin, Ireland.
Helmut Schmid (1994): Probabilistic Part-of-Speech Tagging Using Decision Trees. Proceedings of International Conference on New Methods in Language Processing, Manchester, UK.
Download
Executable code for PC-Linux, Windows, Mac-OS, and ARM-Linux and parameter files for various languages can be downloaded via the links below.
This software is freely available for research, education and evaluation. For commercial and other licenses, please contact the developer via the email address at the bottom of the page.
Please read the license terms, before you download the software! By downloading the software, you agree to the terms stated there.
The following steps are necessary to install the TreeTagger (see below for the Windows version). Download the files by right-clicking on the link. Then select "save file as". All files should be stored in the same directory.
1. Download the tagger package for your system (PC-Linux, Mac OS-X, ARM-Linux).
If you have problems with your Linux kernel version, download this older Linux version and rename it to tree-tagger-linux-3.2.1.tar.gz.
2. Download the tagging scripts into the same directory.
3. Download the installation script install-tagger.sh.
4. Download the parameter files for the languages you want to process.
5. Open a terminal window and run the installation script in the directory where you have downloaded the files:
sh install-tagger.sh
6. Make a test, e.g.
echo 'Hello world!' | cmd/tree-tagger-english
or
echo 'Das ist ein Test.' | cmd/tagger-chunker-german
Make sure that the files are not automatically unzipped i.e. that the file ending .gz is still present. If you have difficulties with the installation, have a look at the installation hints (kindly provided by Joachim Wagner).
Parameter files
Bulgarian parameter file (gzip compressed, UTF-8, tagset documentation, trained on the Bulgarian Treebank)

Das Problem


google.github.io/seq2seq/getting_started/
seq2seq
Search docs
Overview
Getting Started
Download & Setup
Common Installation Issues
Incorrect matploblib backend
Next Steps
Concepts
Tutorial: Neural Machine Translation
Tutorial: Summarization
Tutorial: Image Captioning
Data
Training
Inference
Tools
Results
Getting Help
Contributing
License
Reference: Models
« Previous
Next »
Docs » Getting Started
Download & Setup
To use tf-seq2seq you need a working installation of TensorFlow 1.0 with Python 2.7 or Python 3.5. Follow the TensorFlow Getting Started guide for detailed setup instructions. With TensorFlow installed, you can clone this repository:
git clone https://github.com/google/seq2seq.git
cd seq2seq
Install package and dependencies
pip install -e .
To make sure everything works as expect you can run a simple pipeline unit test:
python -m unittest seq2seq.test.pipeline_test
If you see a "OK" message, you are all set. Note that you may need to install pyrouge, pyyaml, and matplotlib, in order for these tests to pass. If you run into other setup issues, please file a Github issue.
Common Installation Issues
Incorrect matploblib backend
In order to generate plots using matplotlib you need to have set the correct backend. Also see this StackOverflow thread. To use the Agg backend, simply:
The tagger is describ
Helmut Schmi
Helmut Schmi
Processing, Ma
Downloac
Executable code for
This software is freel
the bottom of the pa
Please read the lice
The following steps a
file as". All files shoul
Download the
If you have pr
Download the
Download the
Download the
Open a termin
sh install-tagg
Make a test, e.
echo 'Hello w
or
echo 'Das ist e
Make sure that the files are not automatically unzipped i.e. that the file ending .gz is still present. If you have difficulties with the installation, have a look at the installation hints (kindly provided by Joachim Wagner).
Parameter files
Bulgarian parameter file (gzip compressed, UTF-8, tagset documentation, trained on the Bulgarian Treebank)

Das Problem


google.github.io/seq2seq/getting_started/
seq2seq
Search docs
Overview
Getting Started
Download & Setup
Common Installation Issues
Incorrect matploblib backend
Next Steps
Concepts
Tutorial: Neural Machine Translation
Tutorial: Summarization
Tutorial: Image Captioning
Data
Training
Inference
Tools
Results
Getting Help
Contributing
License
Reference: Models
« Previous
Docs » Getting Started
GitHub, Inc. github.com/tensorflow/models/tree/ma
Manual installation
Running and training SyntaxNet/DRAGNN models requires building this package from source. You'll need to install:
python 2.7:
Python 3 support is not available yet
bazel:
Follow the instructions here
Alternately, Download bazel <.deb> from https://github.com/bazelbuild/bazel/releases for your system configuration.
Install it using the command: sudo dpkg -i <.deb file>
Check for the bazel version by typing: bazel version
swig:
apt-get install swig on Ubuntu
brew install swig on OSX
protocol buffers, with a version supported by TensorFlow:
check your protobuf version with pip freeze | grep protobuf
upgrade to a supported version with pip install -U protobuf==3.3.0
mock, the testing package:
pip install mock
asciitree, to draw parse trees on the console for the demo:
pip install asciitree
numpy, package for scientific computing:
pip install numpy
pygraphviz to visualize traces and parse trees:
apt-get install -y graphviz libgraphviz-dev
pip install pygraphviz --install-option="--include-path=/usr/include/graphviz" --install-option="--library-path=/usr/lib/graphviz/"
Download
Please read the lice
Make sure that the files are not automatically unzipped i.e. that t
installation hints (kindly provided by Joachim Wagner).
Parameter files
Bulgarian parameter file (gzip compressed, UTF-8, tagset documentation, trained on the Bulgarian Treebank)

Manuelles Installieren

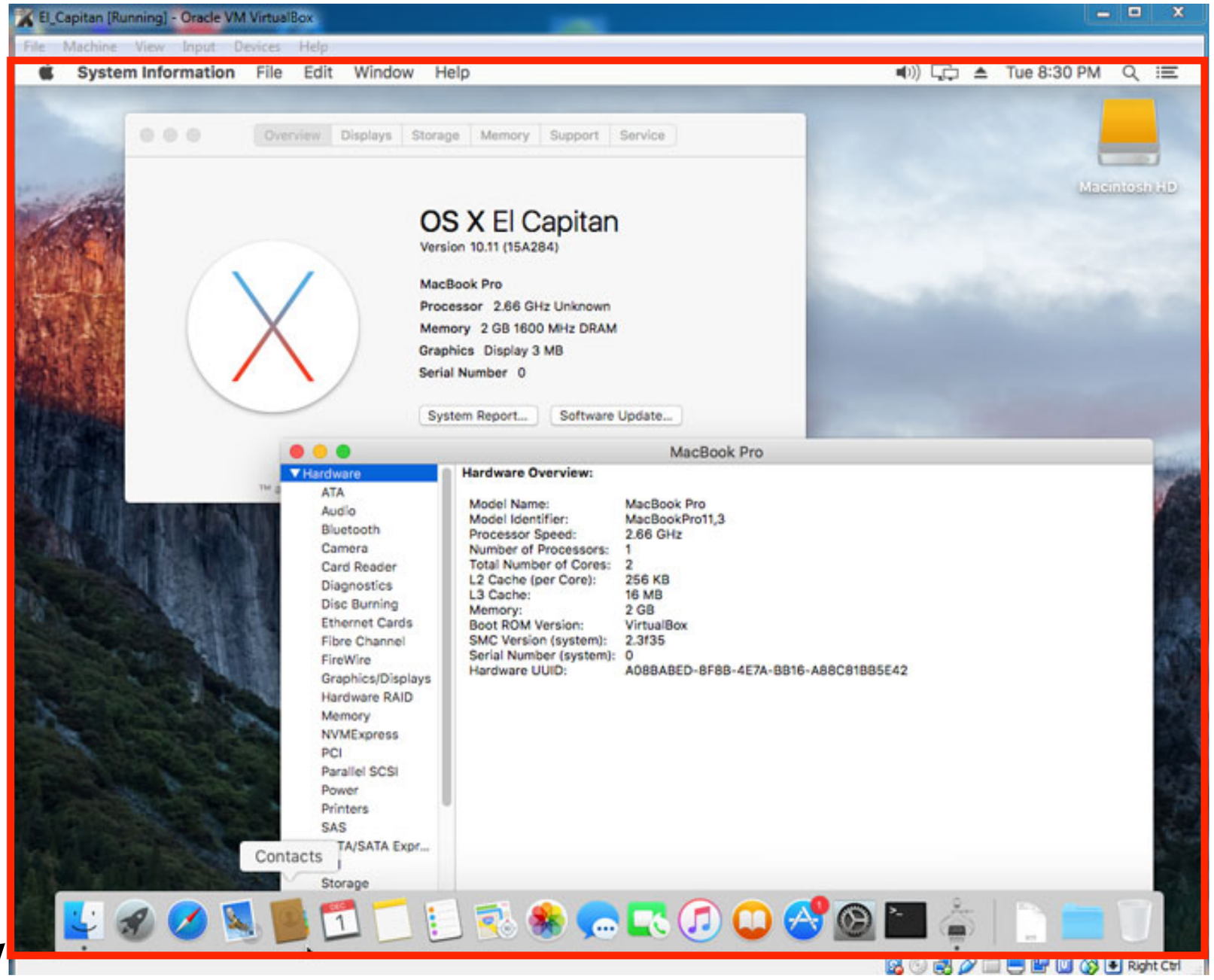
- Jeder Teilnehmer installiert zu Hause erstmal alle notwendigen Bibliotheken und kompiliert das Tool von Hand.

- Sehr zeit- und fehleranfällig.
  - Leute machen ihre Hausaufgaben eher, wenn man sie ihnen sehr, sehr leicht macht.
  - “Ich hatte die Woche über keine Zeit, kann ich die Software nicht einfach jetzt installieren?”
  - “Ich hab die Installationsanweisungen genau befolgt, aber es klappt einfach nicht.”

Virtuelle Maschinen

- Virtuelle Maschinen, z.B. VirtualBox:
  Komplettes Gast-OS wird innerhalb Host-OS gebootet.

Host: Windows

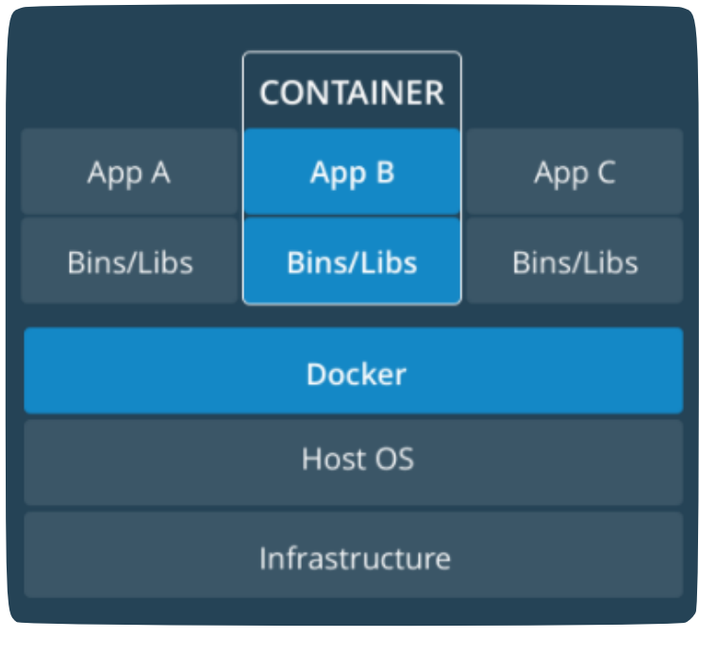


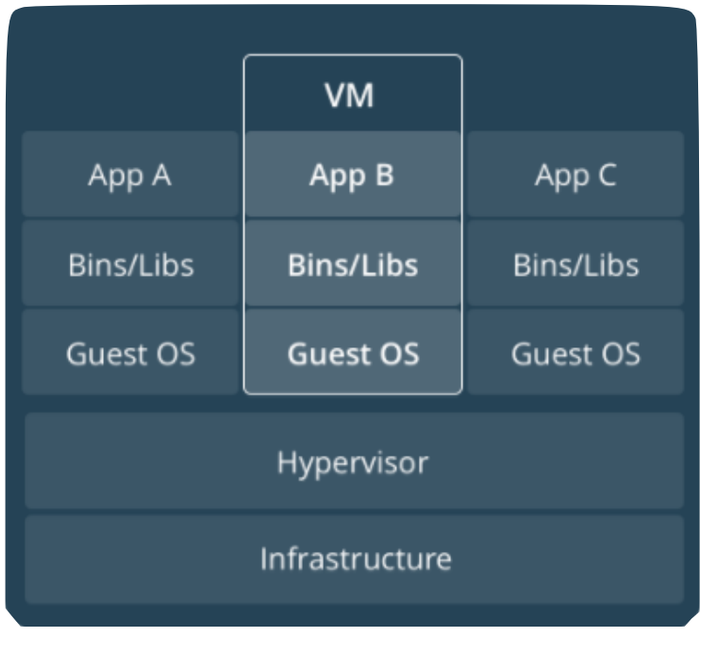
Guest: MacOS

Container

- VMs sind sehr speicher- und bootzeitintensiv und für viele Anwendungen Overkill.

- Container (z.B. Docker): Alle Container teilen sich so viele OS-Ressourcen wie möglich, aber Prozesse und Dateisysteme sind voneinander isoliert.





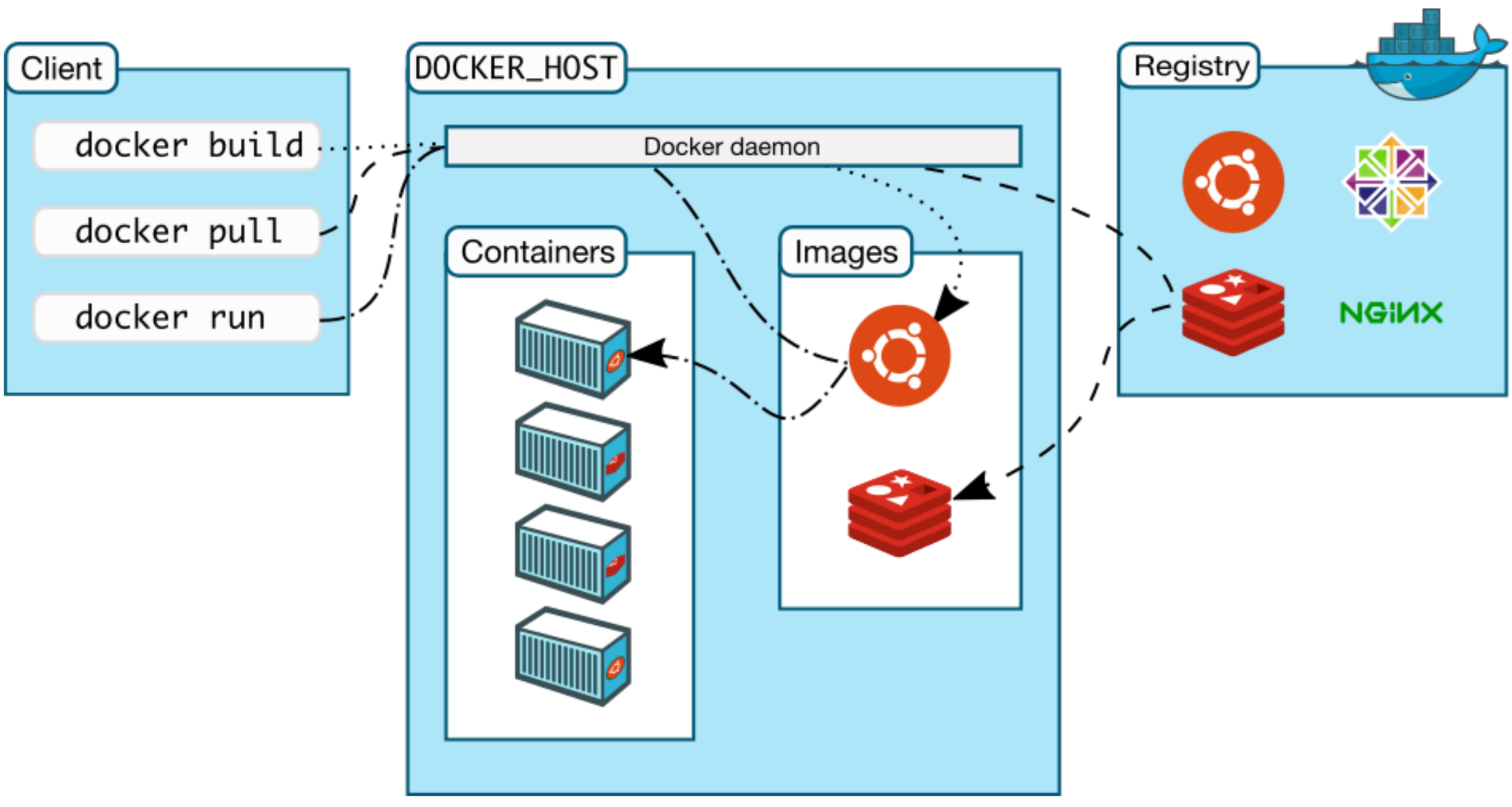
Bestandteile von Docker

- Docker *Image* fixiert Zustand einer “Maschine”, z.B. Inhalt des Dateisystems.

- Docker *Container* startet einen Prozess in einem Image. Dieser Prozess läuft direkt auf Host-OS.
  - kann viele Container mit dem gleichen Image haben

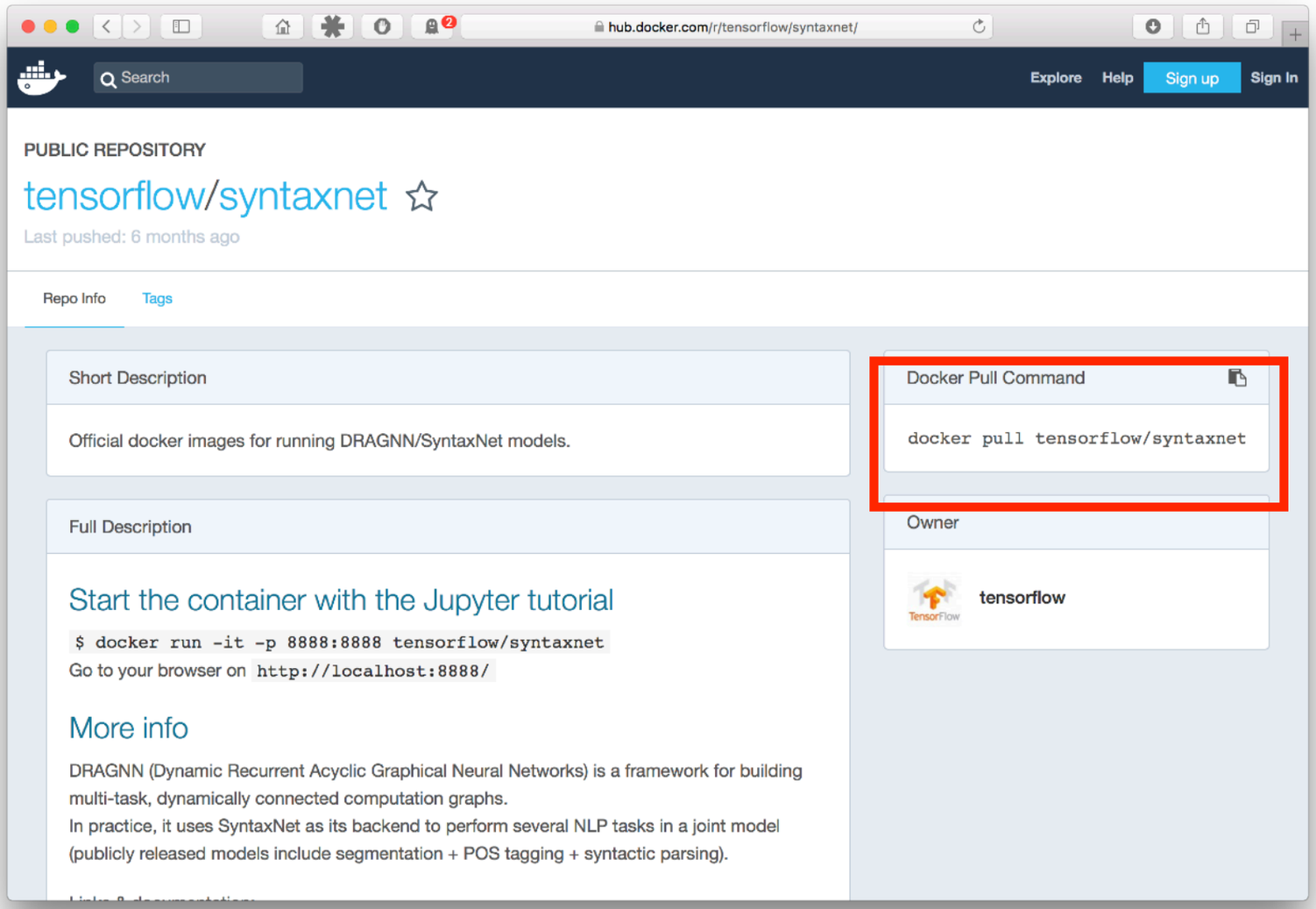
- Docker *Daemon* wird einmal gestartet (z.B. beim Booten der Host-Maschine) und kümmert sich um die Verwaltung der Container.

Bestandteile von Docker



Woher kommen die Images?

- Option 1: Aus einer Docker Registry herunterladen.



- Option 2: Eigenes Dockerfile schreiben.

Beispiel

```
hilbert:~ koller$ docker run ubuntu /bin/echo hello world
hello world
```

Beispiel

Erzeuge und starte
neuen Docker-Container …

```
hilbert:~ koller$ docker run ubuntu /bin/echo hello world
hello world
```

Beispiel

Erzeuge und starte
neuen Docker-Container …

```
hilbert:~ koller$ docker run ubuntu /bin/echo hello world
hello world
```

… auf Grundlage des
Images “ubuntu” …

Beispiel

Erzeuge und starte neuen Docker-Container …

```
hilbert:~ koller$ docker run ubuntu /bin/echo hello world
hello world
```

… auf Grundlage des Images “ubuntu” …

… und führe im Container dieses Kommando aus.

Beispiel: Interaktiver Container

```
hilbert:~ koller$ docker run -i -t ubuntu /bin/bash
root@f4572742f735:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt
opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@f4572742f735:/# echo hello world
hello world
root@f4572742f735:/# exit
exit
hilbert:~ koller$
```

Beispiel: Interaktiver Container

Interaktives Terminal erzeugen und mit Konsole verbinden.

```
hilbert:~ koller$ docker run -i -t ubuntu /bin/bash
root@f4572742f735:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt
opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@f4572742f735:/# echo hello world
hello world
root@f4572742f735:/# exit
exit
hilbert:~ koller$
```

Beispiel: Interaktiver Container

Interaktives Terminal erzeugen und mit Konsole verbinden.

```
hilbert:~ koller$ docker run -i -t ubuntu /bin/bash
root@f4572742f735:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt
opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@f4572742f735:/# echo hello world
hello world
root@f4572742f735:/# exit
exit
hilbert:~ koller$
```

Im Container gibt es komplettes Dateisystem (hier: mit Ubuntu-Distribution).

Beispiel: Interaktiver Container

Interaktives Terminal erzeugen und mit Konsole verbinden.

```
hilbert:~ koller$ docker run -i -t ubuntu /bin/bash
root@f4572742f735:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt
opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@f4572742f735:/# echo hello world
hello world
root@f4572742f735:/# exit
exit
hilbert:~ koller$
```

Im Container gibt es komplettes Dateisystem (hier: mit Ubuntu-Distribution).

Wir sind im Container der Root-User.

Beispiel: ps und images

“docker ps”: Alle laufenden Container anzeigen.

```
hilbert:~ koller$ docker ps
CONTAINER ID   IMAGE      COMMAND       CREATED          STATUS         PORTS    NAMES
0dfe46c1fc5b   ubuntu     "/bin/bash"   10 seconds ago   Up 9 seconds           objective_wozniak
```

“docker images”: Alle installierten Images anzeigen.

```
hilbert:~ koller$ docker images
REPOSITORY                          TAG     IMAGE ID          CREATED       SIZE
ubuntu                              latest  747cb2d60bbe      3 weeks ago   122MB
readingcomprehension_pytorch        latest  51fc0099b296      3 weeks ago   3.36GB
hackathon17_rc                      latest  56d9ffaf6b7b      3 weeks ago   3.25GB
```

Verzeichnisse mounten

- Verzeichnisse des Host-Systems kann man in den Docker-Container einhängen (mounten):

```
hilbert:tokenization koller$ docker run -i -t -v /Users/koller/tokenization:/local ubuntu /bin/bash
root@9bf732c2ce76:/# ls /local
Dockerfile  test.tok  test.txt
```

- Insbesondere wenn Docker-Container direkt ein Programm ausführt, kann man damit fast genauso arbeiten, wie wenn das Programm lokal installiert wäre.

Verzeichnisse mounten

- Verzeichnisse des Host-Systems kann man in den Docker-Container einhängen (mounten):

Dieses Host-Verzeichnis …

```
hilbert:tokenization koller$ docker run -i -t -v /Users/koller/tokenization:/local ubuntu /bin/bash
root@9bf732c2ce76:/# ls /local
Dockerfile  test.tok  test.txt
```

- Insbesondere wenn Docker-Container direkt ein Programm ausführt, kann man damit fast genauso arbeiten, wie wenn das Programm lokal installiert wäre.

Verzeichnisse mounten

- Verzeichnisse des Host-Systems kann man in den Docker-Container einhängen (mounten):

Dieses Host-Verzeichnis …

… unter diesem Pfad im Container mounten.

```
hilbert:tokenization koller$ docker run -i -t -v /Users/koller/tokenization:/local ubuntu /bin/bash
root@9bf732c2ce76:/# ls /local
Dockerfile  test.tok  test.txt
```

- Insbesondere wenn Docker-Container direkt ein Programm ausführt, kann man damit fast genauso arbeiten, wie wenn das Programm lokal installiert wäre.

Dockerfiles

Ein Dockerfile ist eine Textdatei, die spezifiziert,
wie ein Image Schritt für Schritt gebaut werden soll.

```
FROM phusion/baseimage

RUN apt-get update && apt-get install -y wget python git

RUN wget http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/tree-tagger-linux-3.2.1.tar.gz
RUN wget http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/tagger-scripts.tar.gz
RUN wget http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/install-tagger.sh
RUN wget http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/german-par-linux-3.2-utf8.bin.gz

# unpack everything
RUN tar xzvf tree-tagger-linux-3.2.1.tar.gz
RUN tar xzvf tagger-scripts.tar.gz
RUN gunzip german-par-linux-3.2-utf8.bin.gz
RUN ln -s /german-par-linux-3.2-utf8.bin /lib/german-utf8.par
```

Dockerfiles

Ein Dockerfile ist eine Textdatei, die spezifiziert,
wie ein Image Schritt für Schritt gebaut werden soll.

Startpunkt ist dieses Docker-Image

```
FROM phusion/baseimage

RUN apt-get update && apt-get install -y wget python git

RUN wget http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/tree-tagger-linux-3.2.1.tar.gz
RUN wget http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/tagger-scripts.tar.gz
RUN wget http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/install-tagger.sh
RUN wget http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/german-par-linux-3.2-utf8.bin.gz

# unpack everything
RUN tar xzvf tree-tagger-linux-3.2.1.tar.gz
RUN tar xzvf tagger-scripts.tar.gz
RUN gunzip german-par-linux-3.2-utf8.bin.gz
RUN ln -s /german-par-linux-3.2-utf8.bin /lib/german-utf8.par
```

Dockerfiles

Ein Dockerfile ist eine Textdatei, die spezifiziert,
wie ein Image Schritt für Schritt gebaut werden soll.

Startpunkt ist dieses Docker-Image

```
FROM phusion/baseimage

RUN apt-get update && apt-get install -y wget python git

RUN wget http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/tree-tagger-linux-3.2.1.tar.gz
RUN wget http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/tagger-scripts.tar.gz
RUN wget http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/install-tagger.sh
RUN wget http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/german-par-linux-3.2-utf8.bin.gz

# unpack everything
RUN tar xzvf tree-tagger-linux-3.2.1.tar.gz
RUN tar xzvf tagger-scripts.tar.gz
RUN gunzip german-par-linux-3.2-utf8.bin.gz
RUN ln -s /german-par-linux-3.2-utf8.bin /lib/german-utf8.par
```

Dann diese Linux-Kommandos ausführen.

Image aus Dockerfile bauen

Anweisungen im Dockerfile werden einmal ausgeführt.
Endzustand der Maschine wird in Image gespeichert.

```
hilbert:tokenization koller$ docker build -t tokenize_de .
Sending build context to Docker daemon  4.096kB
Step 1/3 : FROM phusion/baseimage
 ---> 877509368a8d
Step 2/3 : RUN apt-get update && apt-get install -y wget perl
 ---> Using cache
 ---> f4a89e779360
Step 3/3 : RUN wget https://www.linguistics.ruhr-uni-bochum.de/~dipper/pub/software/tokenize.perl
 ---> Using cache
 ---> 2e36d5d73f14
Successfully built 2e36d5d73f14
Successfully tagged tokenize_de:latest

hilbert:tokenization koller$ docker images
REPOSITORY                  TAG         IMAGE ID            CREATED             SIZE
tokenize_de                 latest      2e36d5d73f14        3 weeks ago         312MB
```

Image aus Dockerfile bauen

Anweisungen im Dockerfile werden einmal ausgeführt.
Endzustand der Maschine wird in Image gespeichert.

```
hilbert:tokenization koller$ docker build -t tokenize_de .
Sending build context to Docker daemon  4.096kB
Step 1/3 : FROM phusion/baseimage
 ---> 877509368a8d
Step 2/3 : RUN apt-get update && apt-get install -y wget perl
 ---> Using cache
 ---> f4a89e779360
Step 3/3 : RUN wget https://www.linguistics.ruhr-uni-bochum.de/~dipper/pub/software/tokenize.perl
 ---> Using cache
 ---> 2e36d5d73f14
Successfully built 2e36d5d73f14
Successfully tagged tokenize_de:latest

hilbert:tokenization koller$ docker images
REPOSITORY                  TAG         IMAGE ID            CREATED             SIZE
tokenize_de                 latest      2e36d5d73f14        3 weeks ago         312MB
```

Menschenlesbaren Namen zuweisen.

Image aus Dockerfile bauen

Anweisungen im Dockerfile werden einmal ausgeführt.
Endzustand der Maschine wird in Image gespeichert.

```
hilbert:tokenization koller$ docker build -t tokenize_de .
Sending build context to Docker daemon  4.096kB
Step 1/3 : FROM phusion/baseimage
 ---> 877509368a8d
Step 2/3 : RUN apt-get update && apt-get install -y wget perl
 ---> Using cache
 ---> f4a89e779360
Step 3/3 : RUN wget https://www.linguistics.ruhr-uni-bochum.de/~dipper/pub/software/tokenize.perl
 ---> Using cache
 ---> 2e36d5d73f14
Successfully built 2e36d5d73f14
Successfully tagged tokenize_de:latest

hilbert:tokenization koller$ docker images
REPOSITORY                  TAG         IMAGE ID            CREATED             SIZE
tokenize_de                 latest      2e36d5d73f14        3 weeks ago         312MB
```

Menschenlesbaren Namen zuweisen.

Danach existiert Image auf meinem Computer.
Kann mit “docker run” Container erzeugen.

Vorbereitung Ihrer Sitzung

- Manche Tools in diesem Proseminar sind in Java implementiert und damit portabel.
  - Können wir direkt verwenden.
  - Jede/r von Ihnen sollte aktuelle Version von Java installieren: http://www.oracle.com/technetwork/java/javase/downloads
- Für alle anderen Tools sollten Sie ein Dockerfile schreiben und im Piazza teilen.
  - Alle anderen Teilnehmer bitte *vor* der Sitzung herunterladen und mit “docker build” bauen.

Übungen

- Interaktiver Docker-Container für Image “ubuntu”:
  - Starten (interaktiv und nicht-interaktiv) und einige Kommandos ausführen.
  - Lokales Verzeichnis einhängen und Dateien aus Docker heraus lesen/schreiben/ändern.
  - docker images, ps usw. ausprobieren

Übungen

- Evalb: Programm zum Evaluieren von Parsern.
  Aber online nur als 32-bit-Version verfügbar:
  http://nlp.cs.nyu.edu/evalb/

- Bauen Sie ein Dockerfile für Evalb. Verwenden Sie dazu “FROM i686/ubuntu” (für 32-bit Linux).

```
hilbert:evalb koller$ docker run -v <Pfad>:/local evalb /EVALB/evalb /local/ptb.txt /local/ptb.txt

  Sent.                          Matched  Bracket   Cross        Correct Tag
 ID  Len.  Stat. Recal  Prec.  Bracket gold test Bracket Words  Tags Accracy
============================================================================
   1   18     0  100.00 100.00     12     12   12       0     18    18   100.00
   2   13     0  100.00 100.00     10     10   10       0     13    13   100.00
============================================================================
               100.00 100.00     22     22   22       2     31    31   100.00

...
```