Shared Task Introduction

Learning Machine Learning

Nils Reiter

September 26-27, 2018

Outline

Shared Tasks

- Established framework in NLP
- Driver of innovation in the past decade (e.g., machine translation)
- Competitive, winners are highly respected

Shared Tasks

- Established framework in NLP
- Driver of innovation in the past decade (e.g., machine translation)
- Competitive, winners are highly respected
- Past STs
  - Chunking Sang and Buchholz (2000)
  - Clause identification Sang and Déjean (2001)
  - Language-independent named entity recognition Tjong Kim Sang and De Meulder (2003)
  - Syntactic parsing either multilingually or for specific languages Buchholz and Marsi (2006), Kübler (2008), and Nivre et al. (2007)
  - semantic representation/role labeling Bos (2008), Carreras and Màrquez (2004), and Carreras and Màrquez (2005)
  - ...

Shared Tasks

Workflow

- Organizers define a task and provide a data set with annotations
- Participants develop (automatic) systems to solve the task
- $t-2$: Previously unknown test data is published (without annotations), participants apply their systems to this data set
- $t-1$: Participants upload/send the results of their systems to the organizers
- t: Organizers evaluate each system's results against a (secret) gold standard, results are published
- $t+1$: Gold standard is published, papers written, workshops held

Section 2

Data and Annotations

Corpus

Title	Description
Werther	Goethe's *Sorrows of the Young Werther*; pistolary novel, published 1774/1787
Bundestagsdebatten	Debates from the German federal parliament; stenographic minutes
Parzival	Middle High German Arthurian Romance; written 12th/13th century CE; verse

Table: Corpus overview

- Heterogeneous with respect to content and form
- German/Middle High German

Background: Research interests

- Werther (Modern German Literature)
  - Successful novel, a large number of adaptations have been published
  - What makes a Werther adaptation ('Wertheriade') recognizable as an adaptation (e.g., Wertherness?)
    - Three characters in a triadic relationship (Werther, Lotte, Albert)
    - Importance of nature (e.g., certain lakes or forest clearings)

Background: Research interests

- Werther (Modern German Literature)
  - Successful novel, a large number of adaptations have been published
  - What makes a Werther adaptation ('Wertheriade') recognizable as an adaptation (e.g., Wertherness?)
    - Three characters in a triadic relationship (Werther, Lotte, Albert)
    - Importance of nature (e.g., certain lakes or forest clearings)
- Parliamentary debates (Social Sciences)
  - Relation of armed conflicts and identity building
  - Who mentions which institution in what context?

Background: Research interests

- Werther (Modern German Literature)
  - Successful novel, a large number of adaptations have been published
  - What makes a Werther adaptation ('Wertheriade') recognizable as an adaptation (e.g., Wertherness?)
    - Three characters in a triadic relationship (Werther, Lotte, Albert)
    - Importance of nature (e.g., certain lakes or forest clearings)
- Parliamentary debates (Social Sciences)
  - Relation of armed conflicts and identity building
  - Who mentions which institution in what context?
- Parzival
  - 16 volumes, many characters and places
  - Social relations between characters and/or places

Background: Research Interests

Common interest

CRETA works on methods/concepts/workflows that are relevant for multiple disciplines/research questions
In this case: Entities!

- Werther: Characters and locations
- Parliamentary debates: Persons, organizations, locations, dates
- Parzival: Characters and locations

Annotations

Conceptual Overview

Text

Figure: Entity references and entities

Annotations

Conceptual Overview

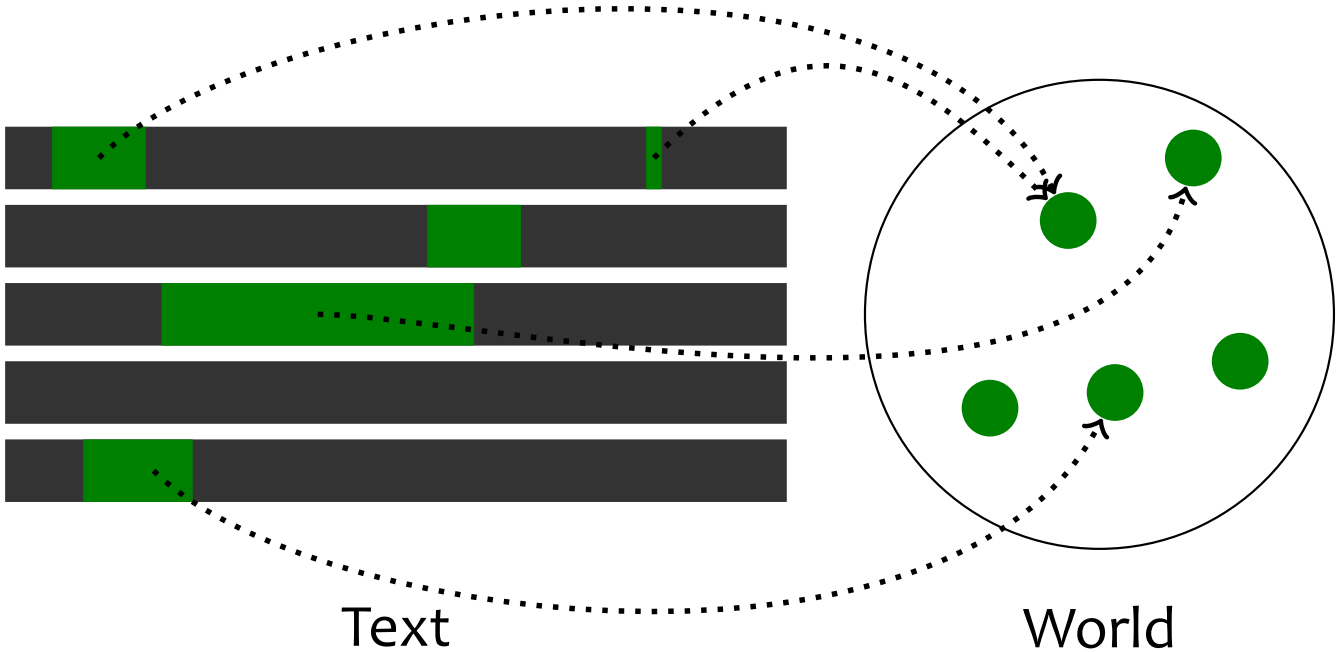


Figure: Entity references and entities

Annotations

Conceptual Overview

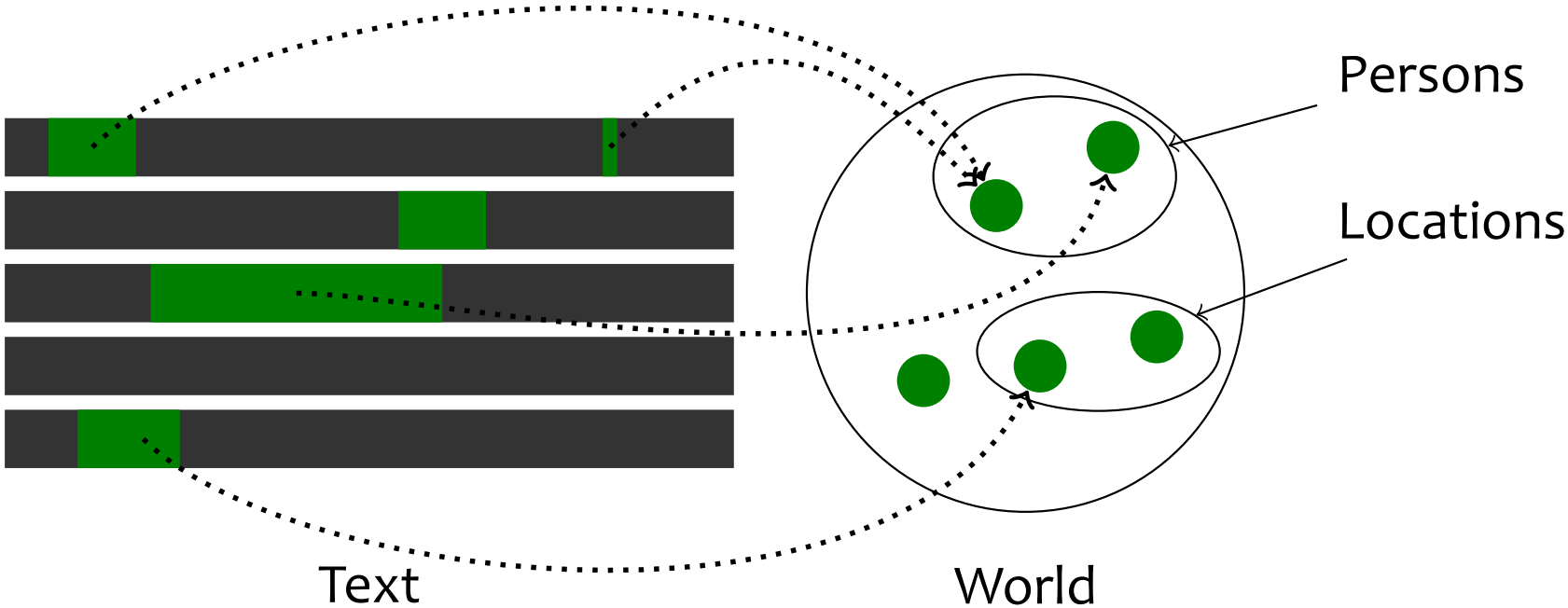


Figure: Entity references and entities

Annotations

Guidelines

Entity references

- ✓ Proper names (‘Werther’)
- ✓ Appellative noun phrases (‘the knight’) – if they refer
  - ✓ Groups: ‘the two knights’
  - ✓ Addresses: ‘My dear friend’
  - ✗ Generic expressions: ‘the chancellor is elected by the parliament’
- ✗ Pronouns are *not annotated*

Annotations

Guidelines

How do we annotate?

- Maximal noun phrases, including
  - relative clauses: 'the chancellor, who has in Berlin at this time'
  - appositions: 'Wilhelm, my friend'
- If determiner and preposition are contracted, the contracted form is included
  - 'in [dem Land]', '[im Land]'
- Embedded phrases are annotated
  - '[Wolfram von [Eschenbach]$_{LOC}$]$_{PER}$'
    - ST data: Only the longest annotation matters
- Entity type is annotated in context
  - 'I always wanted to go to [Europe]$_{LOC}$.'
  - '[Europe]$_{ORG}$ is forcing Greece to take a hard austerity course.'

Annotations

Examples

Text	Classes	Examples
Werther	Person	Werther, liebster Wilhelm, die Kinder aus dem Dorfe
	Location	Die Schweiz, dem Dorfe
	Work	Emilia Galotti
Bundestags-debatten	Person	Angela Merkel, die Abgeordneten
	Location	Großbritannien, das Land, Europa
	Organization	SPD, die Union, Europa
Parzival	Person	Parzival, der ritter, die tafelrunde
	Location	Nantes, der wald Brazilian, der palas

Annotations

Text-specific properties

- Werther
  - 1878: old language
  - Epistolary novel: First-person narrator
  - Emotional style: Long sentences, interjections, …
- Bundestagsdebatten
  - Non-narrative text, logged direct speech
  - Contemporary text: Style and content
- Parzival
  - Middle High German
  - Proper nouns are upper cased
  - Almost all other words are lower cased
  - Segmentation in 30 verses: Each first row upper case

Annotations and Data

Summary

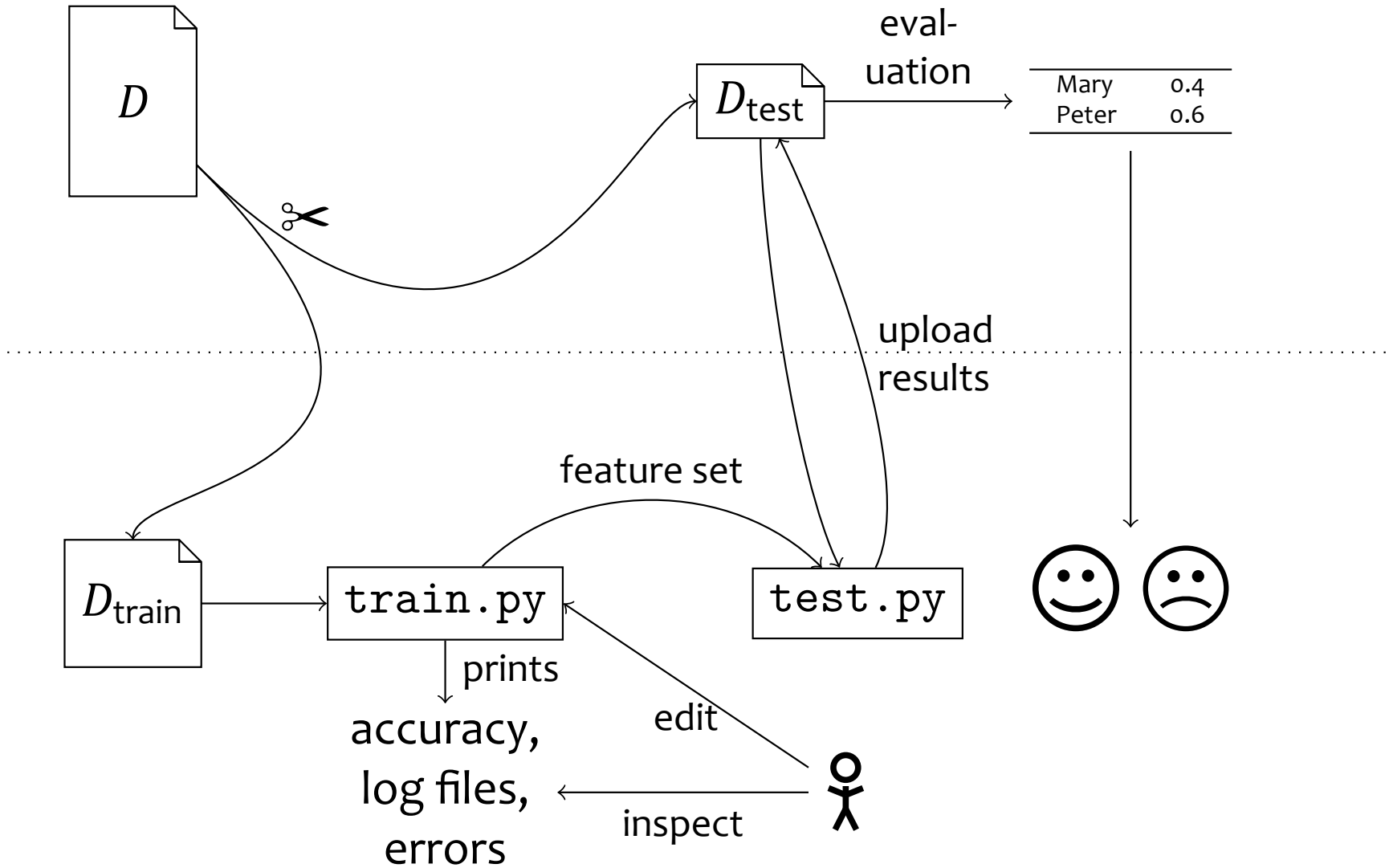
- Three text types with different properties
- Annotated entity references (according to guidelines)
- Files are split into training and test set

Section 3

Hackatorial Setup

Hackatorial

Overview



Hackatorial

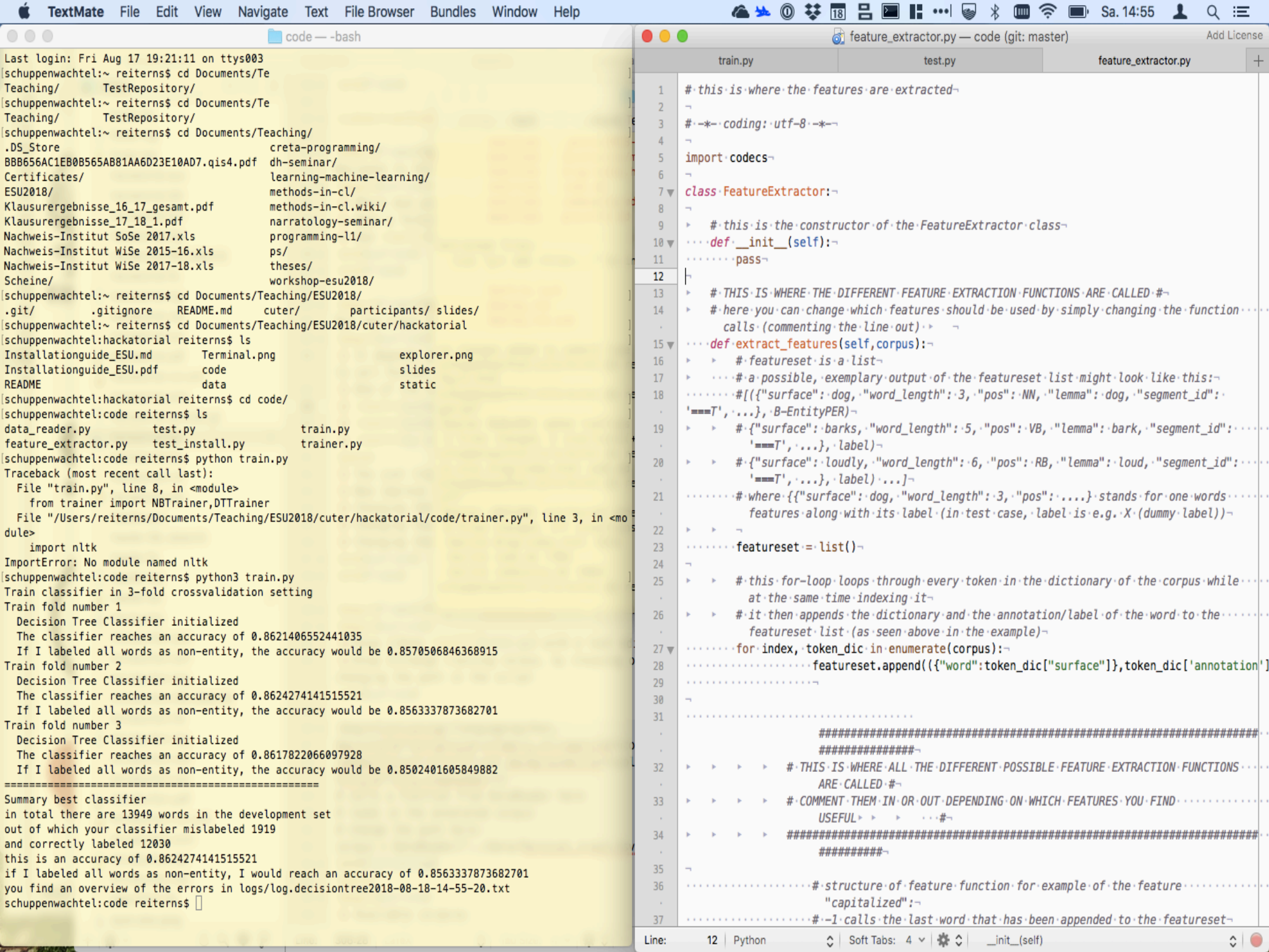
Playground options

- Choose a data set
  - Werther, Parzival, Bundestagsdebatten
- Choose a classifier
  - Decision tree, naive bayes
- Edit the feature set
  - Turn features on/off, add additional features

Hackatorial

Navigate to the correct folder

- Where did you save the `hackatorial` folder?
- Open a Terminal/Eingabeaufforderung
- Use `cd path/to/hackatorial/code` to navigate into the folder


TextMate File Edit View Navigate Text File Browser Bundles Window Help
Sa. 14:55
code — -bash
Last login: Fri Aug 17 19:21:11 on ttys003
schuppenwachtel:~ reiterns$ cd Documents/Te
Teaching/ TestRepository/
schuppenwachtel:~ reiterns$ cd Documents/Te
Teaching/ TestRepository/
schuppenwachtel:~ reiterns$ cd Documents/Teaching/
.DS_Store creta-programming/
BBB656AC1EB0B565AB81AA6D23E10AD7.qis4.pdf dh-seminar/
Certificates/ learning-machine-learning/
ESU2018/ methods-in-cl/
Klausurergebnisse_16_17_gesamt.pdf methods-in-cl.wiki/
Klausurergebnisse_17_18_1.pdf narratology-seminar/
Nachweis-Institut SoSe 2017.xls programming-l1/
Nachweis-Institut WiSe 2015-16.xls ps/
Nachweis-Institut WiSe 2017-18.xls theses/
Scheine/ workshop-esu2018/
schuppenwachtel:~ reiterns$ cd Documents/Teaching/ESU2018/
.git/ .gitignore README.md cuter/ participants/ slides/
schuppenwachtel:~ reiterns$ cd Documents/Teaching/ESU2018/cuter/hackatorial
schuppenwachtel:hackatorial reiterns$ ls
Installationguide_ESU.md Terminal.png explorer.png
Installationguide_ESU.pdf code slides
README data static
schuppenwachtel:hackatorial reiterns$ cd code/
schuppenwachtel:code reiterns$ ls
data_reader.py test.py train.py
feature_extractor.py test_install.py trainer.py
schuppenwachtel:code reiterns$ python train.py
Traceback (most recent call last):
File "train.py", line 8, in <module>
from trainer import NBTrainer,DTTrainer
File "/Users/reiterns/Documents/Teaching/ESU2018/cuter/hackatorial/code/trainer.py", line 3, in <module>
import nltk
ImportError: No module named nltk
schuppenwachtel:code reiterns$ python3 train.py
Train classifier in 3-fold crossvalidation setting
Train fold number 1
Decision Tree Classifier initialized
The classifier reaches an accuracy of 0.8621406552441035
If I labeled all words as non-entity, the accuracy would be 0.8570506846368915
Train fold number 2
Decision Tree Classifier initialized
The classifier reaches an accuracy of 0.8624274141515521
If I labeled all words as non-entity, the accuracy would be 0.8563337873682701
Train fold number 3
Decision Tree Classifier initialized
The classifier reaches an accuracy of 0.8617822066097928
If I labeled all words as non-entity, the accuracy would be 0.8502401605849882
==========
Summary best classifier
in total there are 13949 words in the development set
out of which your classifier mislabeled 1919
and correctly labeled 12030
this is an accuracy of 0.8624274141515521
if I labeled all words as non-entity, I would reach an accuracy of 0.8563337873682701
you find an overview of the errors in logs/log.decisiontree2018-08-18-14-55-20.txt
schuppenwachtel:code reiterns$
feature_extractor.py — code (git: master)
Add License
train.py
test.py
feature_extractor.py
# this is where the features are extracted
# -*- coding: utf-8 -*-
import codecs
class FeatureExtractor:
# this is the constructor of the FeatureExtractor class
def __init__(self):
pass
# THIS IS WHERE THE DIFFERENT FEATURE EXTRACTION FUNCTIONS ARE CALLED #
# here you can change which features should be used by simply changing the function calls (commenting the line out)
def extract_features(self,corpus):
# featureset is a list
# a possible, exemplary output of the featureset list might look like this:
#[{"surface": dog, "word_length": 3, "pos": NN, "lemma": dog, "segment_id": '===T', ...}, B-EntityPER)
# {"surface": barks, "word_length": 5, "pos": VB, "lemma": bark, "segment_id": '===T', ...}, label)
# {"surface": loudly, "word_length": 6, "pos": RB, "lemma": loud, "segment_id": '===T', ...}, label) ...]
# where {{"surface": dog, "word_length": 3, "pos": ....} stands for one words features along with its label (in test case, label is e.g. X (dummy label))
featureset = list()
# this for-loop loops through every token in the dictionary of the corpus while at the same time indexing it
# it then appends the dictionary and the annotation/label of the word to the featureset list (as seen above in the example)
for index, token_dic in enumerate(corpus):
featureset.append(({"word":token_dic["surface"]},token_dic['annotation'])
##########
THIS IS WHERE ALL THE DIFFERENT POSSIBLE FEATURE EXTRACTION FUNCTIONS ARE CALLED
COMMENT THEM IN OR OUT DEPENDING ON WHICH FEATURES YOU FIND USEFUL
##########
#structure of feature function for example of the feature "capitalized":
#-1 calls the last word that has been appended to the featureset
Line: 12 Python Soft Tabs: 4 __init__(self)

Hackatorial

Run the train script using Python

- It depends on your operating system and version, but you can try the following commands to call Python: py, python, python3
- One of the following should work:
  - `python train.py`
  - `python3 train.py`
  - `py train.py`

Hackatorial

Run the train script using Python

- It depends on your operating system and version, but you can try the following commands to call Python: py, python, python3
- One of the following should work:
  - `python train.py`
  - `python3 train.py`
  - `py train.py`
- You just trained your first machine learning model!
- Now improve its performance by
  - Changing the data set
  - Changing the algorithm
  - Changing the feature set

Hackatorial

How to change the data set

Step 1 Open `train.py` with a text editor (e.g. Notepad++)

Step 2 Change training corpus, by choosing one of the available corpora listed below and changing the path in the script

```
# calls a function from DataReader here
# reads in the annotated corpus
# change the path here:
corpus = DataReader("../data/Parzival_train.tsv").read_corpus()
```

Hackatorial

How to change the data set

Step 1 Open `train.py` with a text editor (e.g. Notepad++)

Step 2 Change training corpus, by choosing one of the available corpora listed below and changing the path in the script

```
# calls a function from DataReader here
# reads in the annotated corpus
# change the path here:
corpus = DataReader("../data/Parzival_train.tsv").read_corpus()
```

- Available corpora:
  - `Parzival_train.tsv`
  - `Werther_train.tsv`
  - `Bundestag_train.tsv`

Hackatorial

How to change the features

Step 1 Open `feature_extractor.py` with a text editor

Step 2 Comment or uncomment the features

- Commenting out (disable): Putting a # in front of the line
- Uncomment (enable the feature): Removing the #

```
...
#############################################################################
# THIS IS WHERE ALL THE DIFFERENT POSSIBLE FEATURE EXTRACTION FUNCTIONS ARE CALLED #
# COMMENT THEM IN OR OUT DEPENDING ON WHICH FEATURES YOU FIND USEFUL              #
#############################################################################

# structure of feature function for example of the feature "capitalized":
# -1 calls the last word that has been appended to the featureset
# 0 accesses the dictionary which is the first element of the tupel
# "capitalized" is the feature name

featureset[-1][0]["pos"] = self.pos(token_dic)
#featureset[-1][0]["surface"] = self.surface(token_dic)
#featureset[-1][0]["surface_backwards"] = self.surface_backwards(token_dic)
...
```

The full feature list is available as a PDF (with examples).

Hackatorial

What do features mean?

Available features and their meaning are listed in the table that you got on paper and further below in `feature_extractor.py`

```
    ######################################################################
    # THESE ARE ALL THE DIFFERENT POSSIBLE FEATURE EXTRACTION FUNCTIONS #
    ######################################################################

    # This function returns the part of speech tag of the word
    def pos(self, word_dic):
        return word_dic["pos"]


    # This function returns the word itself
    def surface(self, word_dic):
        return word_dic["surface"]

    # This function returns the word backwards
    def surface_backwards(self, word_dic):
        return word_dic["surface"][::-1]
    ...
```

Hackatorial

How to change the training algorithm

Step 1 Open `train.py` with a text editor

Step 2 Comment out one of the lines starting with `trainer =`

```
    # THIS IS WHERE YOU CAN CHANGE THE ML ALGORITHM#
    # change this line for another ML algorithm (remove the # infront of a line to uncomment)
    # DTTrainer is the trainer for a Decision Tree classifier
    # NBTrainer is the trainer for a Naive Bayes classifier
    #
    trainer = DTTrainer(traincv)
    #trainer = NBTrainer(traincv)
```

Enjoy Training!

References I

Bos, Johan. "Introduction to the Shared Task on Comparing Semantic Representations". In: *Semantics in Text Processing. STEP 2008 Conference Proceedings*. Ed. by Johan Bos and Rodolfo Delmonte. Vol. 1. Research in Computational Semantics. College Publications, 2008, pp. 257–261.

Buchholz, Sabine and Erwin Marsi. "CoNLL-X Shared Task on Multilingual Dependency Parsing". In: *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*. New York City: Association for Computational Linguistics, June 2006, pp. 149–164.

Carreras, Xavier and Lluís Màrquez. "Introduction to the CoNLL-2004 Shared Task: Semantic Role Labeling". In: *HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*. Ed. by Hwee Tou Ng and Ellen Riloff. Boston, Massachusetts, USA: Association for Computational Linguistics, May 2004, pp. 89–97.

References II

Carreras, Xavier and Lluís Màrquez. "Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling". In: *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*. Ann Arbor, Michigan: Association for Computational Linguistics, June 2005, pp. 152–164.

Kübler, Sandra. "The PaGe 2008 Shared Task on Parsing German". In: *Proceedings of the Workshop on Parsing German*. Columbus, Ohio: Association for Computational Linguistics, June 2008, pp. 55–63.

Nivre, Joakim, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. "The CoNLL 2007 Shared Task on Dependency Parsing". In: *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*. Prague, Czech Republic: Association for Computational Linguistics, June 2007, pp. 915–932.

References III

- Sang, Erik F. Tjong Kim and Sabine Buchholz. "Introduction to the CoNLL-2000 Shared Task: Chunking". In: *Proceedings of Fourth Conference on Computational Natural Language Learning and of the Second Learning Language in Logic Workshop*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2000.
- Sang, Erik F. Tjong Kim and Hervé Déjean. "Introduction to the CoNLL-2001 shared task: clause identification". In: *Proceedings of the ACL 2001 Workshop on Computational Natural Language Learning (ConLL)*. 2001.
- Tjong Kim Sang, Erik F. and Fien De Meulder. "Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*. Ed. by Walter Daelemans and Miles Osborne. 2003, pp. 142–147.