Shared Task Introduction
Learning Machine Learning

Nils Reiter

CRETA
CENTER FOR REFLECTED TEXT ANALYTICS

September 26-27, 2018
Outline

Shared Tasks

Data and Annotations

Hackatorial Setup

Concrete steps
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
  - Clause identification
  - Language-independent named entity recognition
  - Syntactic parsing either multilingually or for specific languages
  - Semantic representation/role labeling
- ...
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

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Werther</td>
<td>Goethe’s <em>Sorrows of the Young Werther</em>; pistolary novel, published 1774/1787</td>
</tr>
<tr>
<td>Bundestagsdebatten</td>
<td>Debates from the German federal parliament; stenographic minutes</td>
</tr>
<tr>
<td>Parzival</td>
<td>Middle High German Arthurian Romance; written 12th/13th century CE; verse</td>
</tr>
</tbody>
</table>

**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

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]\textsc{loc}]\textsc{per}’
    ➤ ST data: Only the longest annotation matters
➤ Entity type is annotated in context
  ➤ ‘I always wanted to go to [Europe]\textsc{loc}.’
  ➤ ‘[Europe]\textsc{org} is forcing Greece to take a hard austerity course.’
### Annotations

#### Examples

<table>
<thead>
<tr>
<th>Text</th>
<th>Classes</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Werther</td>
<td>Person</td>
<td>Werther, liebster Wilhelm, die Kinder aus dem Dorfe</td>
</tr>
<tr>
<td></td>
<td>Location</td>
<td>Die Schweiz, dem Dorfe</td>
</tr>
<tr>
<td></td>
<td>Work</td>
<td>Emilia Galotti</td>
</tr>
<tr>
<td>Bundestagsdebatten</td>
<td>Person</td>
<td>Angela Merkel, die Abgeordneten</td>
</tr>
<tr>
<td></td>
<td>Location</td>
<td>Großbritannien, das Land, Europa</td>
</tr>
<tr>
<td></td>
<td>Organization</td>
<td>SPD, die Union, Europa</td>
</tr>
<tr>
<td>Parzival</td>
<td>Person</td>
<td>Parzival, der ritter, die tafelrunde</td>
</tr>
<tr>
<td></td>
<td>Location</td>
<td>Nantes, der wald Brazilian, der palas</td>
</tr>
</tbody>
</table>
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

\[ D \rightarrow D_{\text{test}} \rightarrow \text{evaluation} \]

\[ \text{Mary 0.4} \]
\[ \text{Peter 0.6} \]

\[ D_{\text{train}} \rightarrow \text{train.py} \rightarrow \text{test.py} \]

feature set

upload results

evaluates results

accuracy, log files, errors

edit, inspect

Nils Reiter (CRETA)
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
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 might look like this:
        # [("surface": dog, "word_length": 3, "pos": NN, "lemma": dog, "segment_id": "T", ...)]
        featureset = list()
        for index, token_dic in enumerate(corpus):
            featureset.append([("word": token_dic["surface"]), token_dic["annotation"]])

    # structure of feature function for example of the feature "capitalized":
    # I-1 calls the last word that has been appended to the featureset
    # ...
Hackatorial Setup

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

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

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

```python
# 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#

```python
# 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

```python
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# 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 =`
Enjoy Training!
References


References II


References III

