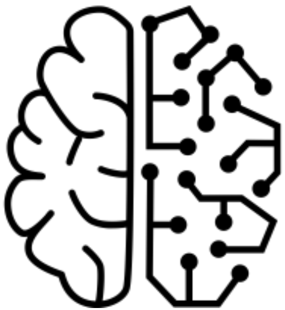


Machine learning for rhetorical figure detection

Felix Klement & Viktoria Schmid

Agenda

- Characteristics of rethorical figures
- Extract potential candidates and features
- Selection of classifier
- Generation of tranings data
- Results
- Conclusion

Characteristics of rethorical figures

Alliteration

Adnomination

"Dan's dog dove deep in the dam, drinking dirty water as he dove."

"He is nobody from nowhere and he knows nothing."

Extract potential candidates and features

1. <u>Selection of the corpora</u>

Problems:

- Nearly no good german sources
- Multiple formats
- No good generic search engines to find texts

Solutions:

- Stick with those provided by nltk
- Train with the Guttenberg Corpus “austen-emma.txt”

Extract potential candidates and features

2. <u>Preprocessing</u>

Problems:

- How to determine a “good” sentence from a “bad” one
- What metrics should be gained

Solutions:

- Use test functions
- Creative selection of some typical text metrics (e.g. distance, similarity, type, levenshtein …)

Extract potential candidates and features

2. Preprocessing

test_alliteration

test_adnomination

gen_word_distance_list

get_dominate_word_type

get_word_distance

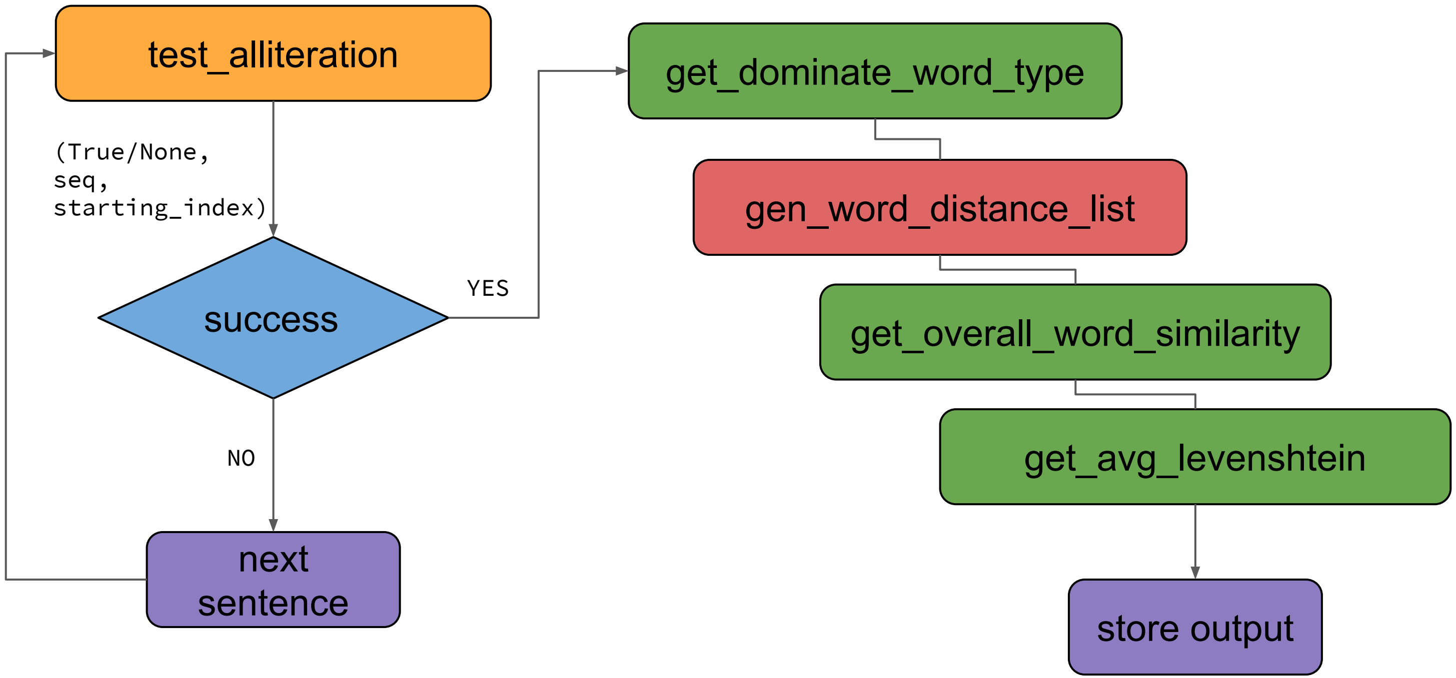
get_avg_word_distance

get_overall_word_similarity

get_avg_levenshtein

Extract potential candidates and features

3. Feature function chaining



Extract potential candidates and features

4. <u>Data organisation/generation</u>

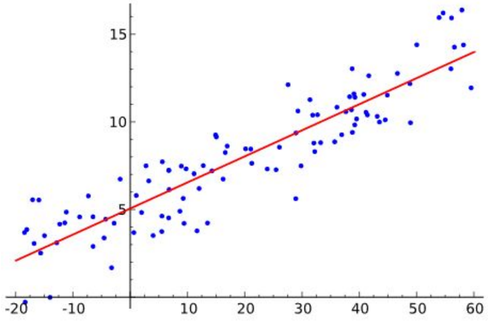
Example of a alliteration tuple:

count	word type	dist	similarity	levenshtein
4	1	1	0.0592592592592592 55	6.0

- <u>count:</u> Words with the same start
- <u>word type:</u> Maximum of repeated word type *('VB', 'NN', 'AT', ...)*
- <u>dist:</u> Distance from one word to the next "important" word
- <u>similarity:</u> Overall word similarity based on *synset wup_similarity*
- <u>levenshtein:</u> The popular Levenshtein distance

Selection of classifier

1. <u>Logistic Regression</u>

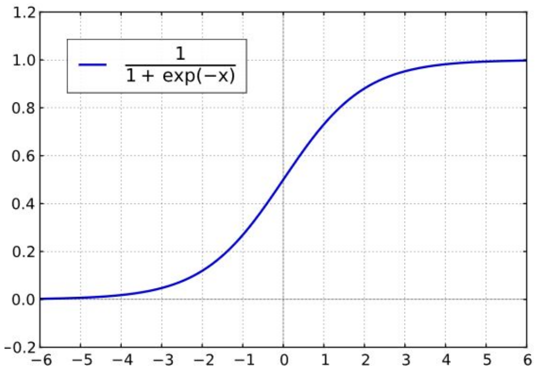


- data free of missing values
- predicting variable is binary or ordinal
- predictors independent from each other

Logistic Regression > Linear regression

both: show influence of each feature on result

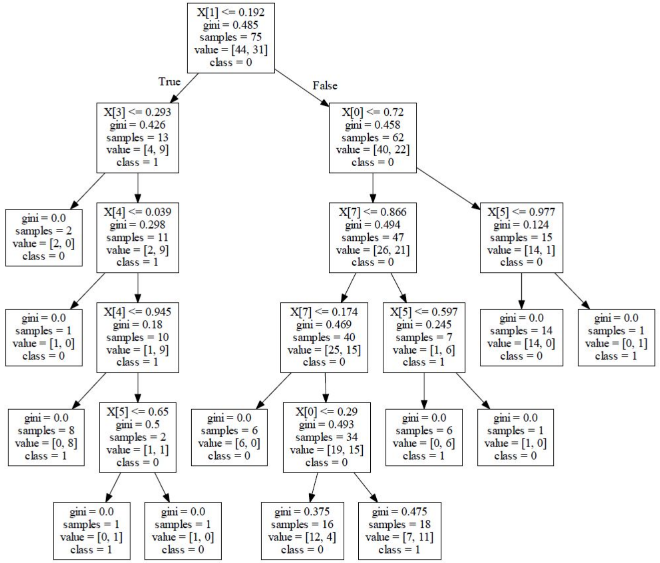
but: feature behaviour were more aligned with sigmoid function



Selection of classifier

2. <u>Decision Tree Classifier</u>

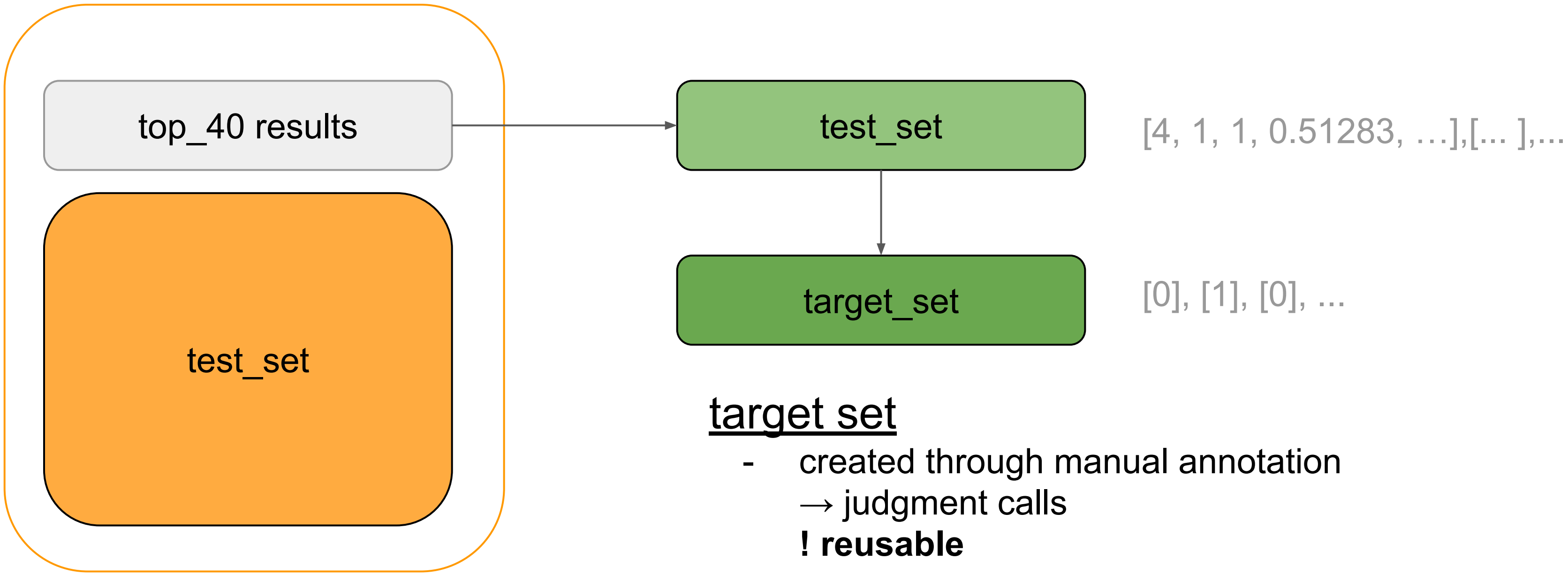
- predicts binary ANDOR multiclass (here: rethorical device)
- allows multiple classifications based on similar features



Advantages:
- accommodates multiple schemes
- easy to follow decision tree

Disadvantages:
- increase in features → decrease in performance
- increase in features → importance of selection order

Generation of trainings data



<u>target set</u>

- created through manual annotation
  → judgment calls
  ! reusable

`for each`
newRethoricalScheme.txt

Results

1. <u>Reviewed Operations</u>

   - `model.score` score

   - `model.predict` prediction prediction

   - `model.predict_prob` probability

```
Out[6]: array([1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0,
               1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1,
               1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0,
               0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0,
               0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1,
               0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1,
               1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1,
               1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1,
               0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1,
               1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1,
               1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1,
               0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1,
               0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1,
               1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1,
               0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1,
               0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
               1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
               1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
               0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0])
```

```
In [7]: tree_model_al.predict_proba(test_al)

Out[7]: array([[0. , 1. ],
               [1. , 0. ],
               [1. , 0. ],
               [1. , 0. ],
               [1. , 0. ],
               [1. , 0. ],
               [1. , 0. ],
               [0.5, 0.5],
```

```
In [10]: reg_model_al.predict_proba(test_al

Out[10]: array([[0.57138704, 0.42861296],
                [0.62562396, 0.37437604],
                [0.62562396, 0.37437604],
                [0.55198214, 0.44801786],
                [0.49567669, 0.50432331],
                [0.55751147, 0.44248853],
                [0.55751147, 0.44248853],
                [0.41578677, 0.58421323],
```

Results

2. Observations

Alliteration

- common scheme when pattern matches
- in Regression Model: often 0.3 - 0.6 probability of being match

Adnomination

- needle-in-a-haystack problem
- in Regression Model: typically 0.8 - 0.9 probability of no match

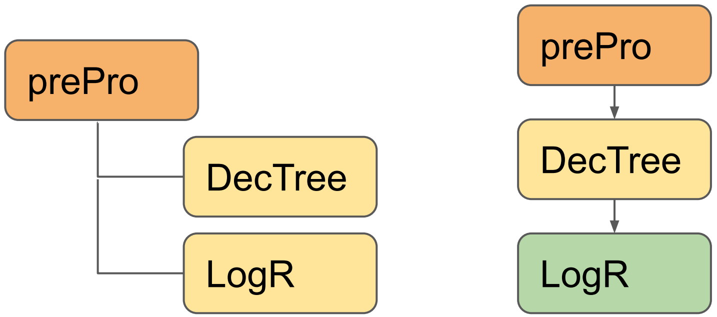
Tuning

- especially interesting for decision tree selection algorithm

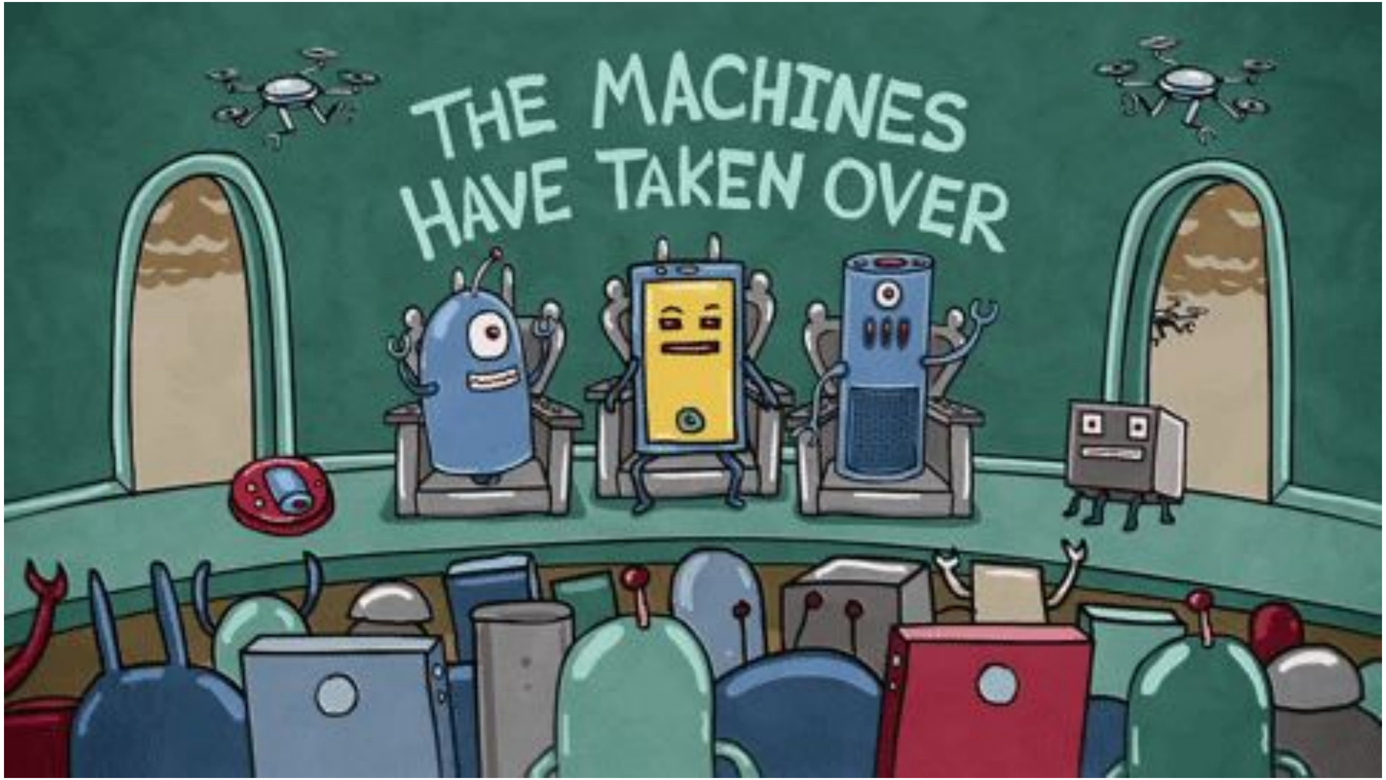
```
In [34]: tree_model_al2 = tree.DecisionTreeClassifier(criterion='entropy')
         tree_model_al2.fit(training_al,target_al)

Out[34]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                     max_features=None, max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                     splitter='best')

In [35]: tree_model_al2.score(training_al,target_al)

Out[35]: 0.95
```

<u>other tuning options:</u>

- improved: target set
- cross-validation
- improved feature selection
- Stacking of machine learning algorithms



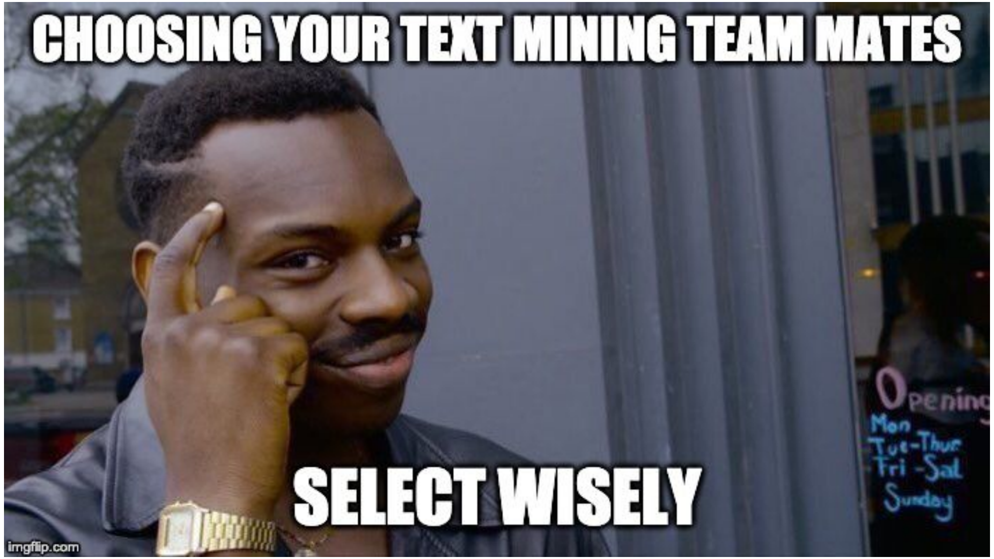
Conclusion



Conclusion

- surface look on how rhetorical figures/schemes
- mining and analysing with the help of nltk toolkit/scikit-learn lib
- technical execution without specialised knowledge about data is limited

Conclusion


CHOOSING YOUR TEXT MINING TEAM MATES
SELECT WISELY
Opening
Mon
Tue-Thur
Fri -Sat
Sunday
imgflip.com

Q & A