Semantic Argument Classification

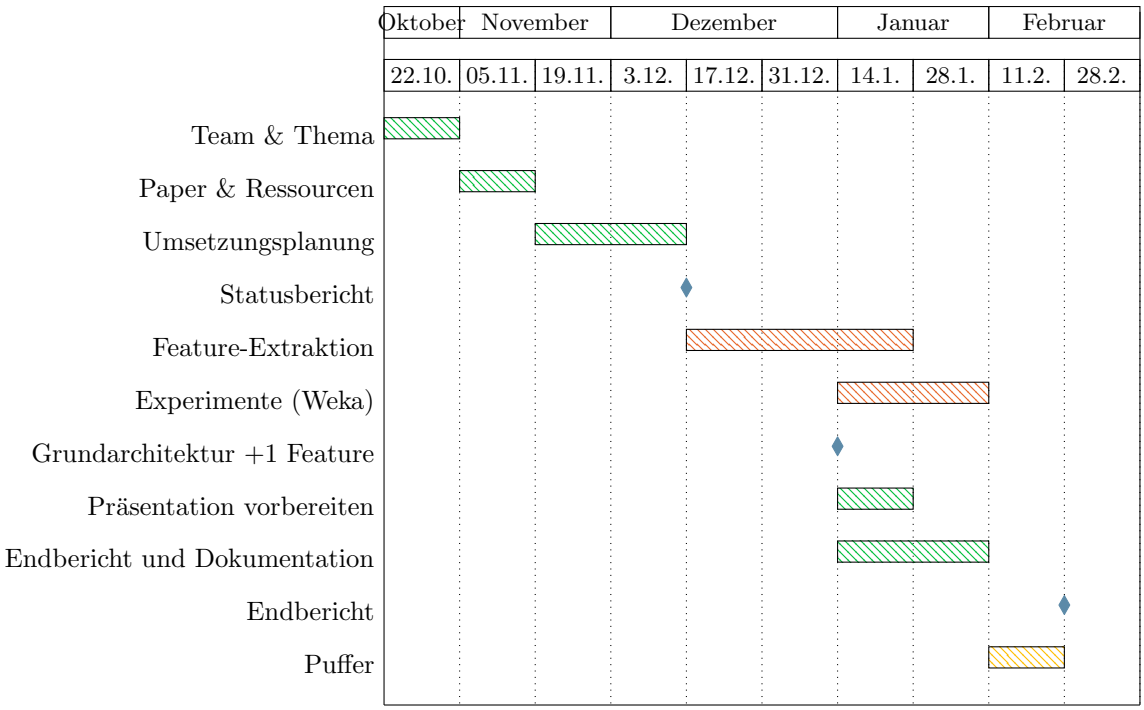
Julian Baumann, Kevin Decker, Maximilian Müller-Eberstein

Ruprecht-Karls-Universität Heidelberg

1 Einleitung

Wem ist durch *wen*, *was*, *wo*, *wie*, *wann* widerfahren? Genau diese Fragen beantwortet die semantische Argumentklassifikation. Den Argumenten des Verbs ihre semantischen Rollen zuzuweisen ist für das tiefere Verständnis textueller Daten unumgänglich. Um diese Aufgabe bestmöglich zu lösen, testen wir verschiedene Klassifikationsalgorithmen und orientieren uns dabei an *Support Vector Learning for Semantic Argument Classication* [1].

1.1 Organisatorisches



2 Grundlagen

TODO: Kevin

2.1 Daten

PropBank in NLTK (Penn Treebank) (Größe)
112917 PropBank-Instanzen/Sätze
292975 Argumente

2.2 Tools

Python 3.4
NLTK 3.0.0
Weka 3.7.11

2.3 Algorithmen

Classifiers
SVM
NaiveBayes
J48

2.4 Vergleichsgrundlagen

Paper: Support Vector Learning for Semantic Argument Classication
PropBanks goldene Annotierung
Trainings- und Testdaten aufsplitten (60% Training, 20% Test, 20% Development)

3 Hauptteil

3.1 Zielsetzung

Unser Ziel ist es Verbargumenten ihre semantischen Rollen zuzuweisen. Wir orientieren uns am PropBank Annotationsschema und wollen die Argumente in ARG[0-4] bzw. ARGM einteilen. Unsere Instanzen sind dabei die Gold Argumente des PropBank Penn Treebank Korpus

ARG0	agent
ARG1	patient
ARG2	instrument, benefactive, attribute
ARG3	starting point, benefactive, attribute
ARG4	ending point
ARGM	modifier

3.2 Umsetzung

Wir extrahieren die Features und die Klasse aus unseren Instanzen des ProbBank Korpus wie folgt:

```
featureList = [...] # zu extrahierende Features
for pbInstance in pbInstances :
    for pbArg in pbInstance.arguments :
        features = []
        for feature in featureList :
            featureList.append(extFeature(feature, pbArg, pbInstance))
# write features to file in ARFF
```

Features

- predicate : nominal
- path : nominal
- phrase type : nominal
- position(before/after) : boolean
- voice(active/passive) : boolean
- headword : nominal
- subcategorization : nominal

siehe Support Vector Learning for Semantic Argument Classication [1]

3.3 Evaluation

Um die Effizienz und Präzision der angewandten Methoden zu ermitteln, erfolgt im Anschluss zu Feature-Extraktion und Experimenten eine Evaluation in zwei Stufen.
Zunächst werden SVM, NaiveBayes und J48 gegeneinander in Hinsicht auf Precision, Recall und F1-Measure verglichen. Da PropBank bereits golden annotiert ist, muss keine weitere manuelle Annotation als Vergleichsgrundlage erstellt werden. Weiterhin kann die Feature-Auswahl für jeden Algorithmus hinsichtlich ihrer Effizienz optimiert werden.
Optional können insbesondere die Ergebnisse des SVM-Verfahrens gegen die des Papers *Support Vector Learning for Semantic Argument Classication* [1] verglichen werden.

4 Ausblick

Im nächsten geplanten Schritt wird die Feature-Extraktion durchgeführt. Die Funktionsvielfalt des NLTK wird hierbei äußerst hilfreich sein. Die anschließenden Experimente in Weka werden auf Grund der verschiedenen Algorithmen und ihrer jeweiligen Feature-Optimierung zeitaufwändig, dank Wekas übersichtlichen Workflow jedoch gut umzusetzen sein.
Wir sind auf die Ergebnisse gespannt und hoffen, dass sie die Auswahl der effizientesten Methode zur semantischen Argumentklassifizierung in Zukunft erleichtern werden.

5 Literatur

Literatur

1. Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, JamesH. Martin, and Daniel Jurafsky. Support vector learning for semantic argument classification. *Machine Learning*, 60(1-3):11–39, 2005.