

Semantic Argument Classification

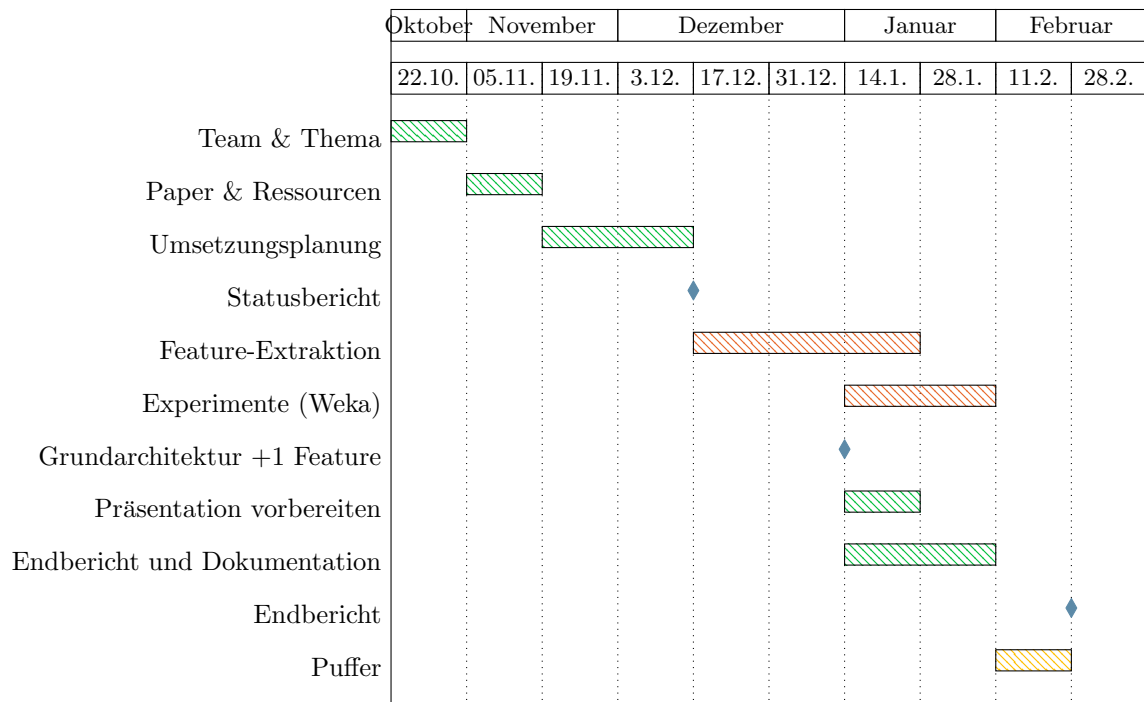
Julian Baumann, Kevin Decker, Maximilian Müller-Eberstein

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 Classification* [2].

1.1 Organisatorisches



2 Grundlagen

2.1 Daten

Als Grundlage für die oben beschriebenen Experimente wird das PropBank-Korpus aus dem Natural Language Toolkit verwendet. Das ursprüngliche Korpus von Martha Palmer

et al. [1] umfasst eine größere Anzahl an PropBank-Instanzen, das im NLTK verwendete besteht allerdings nur aus 112.917 Sätzen. Jedem dieser Sätze wurden seine entsprechenden Argumente durch Annotatoren zugewiesen. Da die meisten Sätze mindestens zwei Argumente besitzen (Subjekt und direktes Objekt) umfasst PropBank 292.975 annotierte Argumente. Diese Daten dienen als Goldstandard für die anschließende Evaluation des Verfahrens.

2.2 Tools

Zur Datenverarbeitung und Auswertung wird die Programmiersprache Python verwendet in der Version 3.4, um die Feature-Extraktion durchzuführen. Durch die Wahl von Python bietet sich das NLTK als Ressource für Sprachverarbeitung an, entsprechend in der Version 3.0.0. Als Machine Learning Tool, mit dem die unten aufgeführten Algorithmen verwendet werden, wird Weka 3.7.11 eingesetzt.

2.3 Algorithmen

Im Laufe des Experiments werden drei verschiedene Klassifizierer eingesetzt, um das bestmögliche Ergebnis in der Argumentklassifikation zu erreichen: eine Support Vector Machine, ein NaiveBayes-Klassifikator und J48, eine OpenSource Java-Implementierung des C4.5-Decision-Tree-Algorithmus. Diese werden dann in der Evaluation gegeneinander verglichen um festzustellen, welcher Klassifizierer das beste Ergebnis erzielt.

2.4 Vergleichsgrundlagen

Um einen Vergleich der Algorithmen ziehen zu können, werden verschiedene Vergleichsgrundlagen verwendet. Die Daten aus dem PropBank-Korpus werden zunächst in Trainings-, Test- und Entwicklungsset aufgeteilt. Da diese bereits annotiert wurden, dienen sie somit als Goldstandard, der zur Berechnung von Precision, Recall und F1-Measure verwendet wird unter Verwendung der verschiedenen Klassifizierer. Zusätzlich können die Ergebnisse aus dem Paper *Support Vector Learning for Semantic Argument Classification* von Pradhan et al. mit denen des SVM-Klassifikators verglichen werden.

3 Hauptteil

3.1 Zielsetzung

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

3.2 Umsetzung

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

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

```

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 [2]

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 Classification* [2] 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.

Literatur

1. Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Comput. Linguist.*, 31(1):71–106, March 2005.
2. 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.