Sehr guter Statusreport.
Aufbau und Formulierung sehr gut.
Einige wenige Rechtschreibfehler und ergänzungswürdige Details.

Semantic Argument Classification

Julian Baumann, Kevin Decker, Maximilian Müller-Eberstein

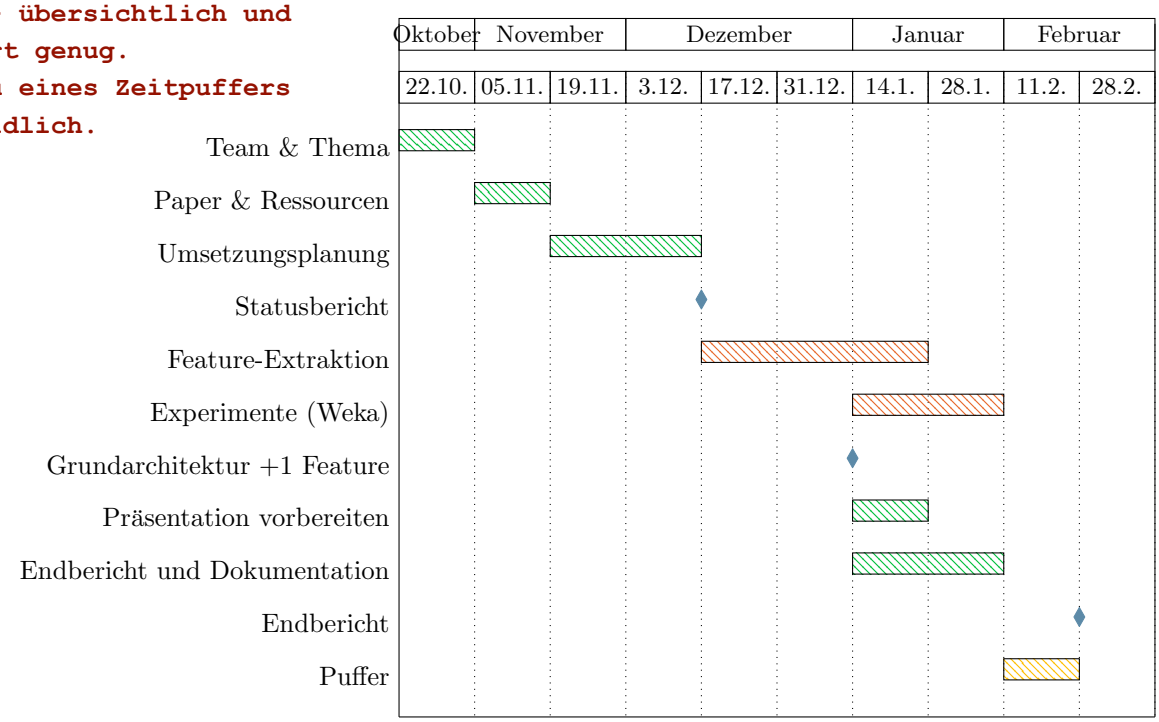
Ruprecht-Karls-Universität Heidelberg

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* [2].

1.1 Organisatorisches

Sehr gut - übersichtlich und detailliert genug.
Der Einbau eines Zeitpuffers ist vorbildlich.



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

Die URL für NLTK-Propbank könnte man angeben.

et al. [1] umfasst eine größere Anzahl an PropBank-Instanzen, das im NLTK verwendete besteht allerdings nur aus 112.917 Sätzen. Jedem dieser Stze 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.

Sätze

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.

(kein Komma nach "Python")

(Komma nach "Tool")

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-Klassifkator 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 Classication* von Pradhan et al. mit denen des SVM-Klassifkator verglichen werden.

-Klassifikators

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 ARG[0-4] bzw. ARGM einteilen. Unsere Instanzen sind dabei die Gold Argumente des PropBank Penn Treebank Korpus.

Es ist für mich nicht eindeutig, wie viele Klassen ihr habt - es könnte zwei sein (ARG[0-4] bzw. ARGM), oder aber sechs. Nach der angegebenen Tabelle ist die zweite Interpretation wahrscheinlicher.

3.2 Umsetzung

Wir extrahieren die Features und die Klasse aus unseren Instanzen des ProbBank 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
```

zum Pseudo-Code siehe später

Features

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

Eine kurze Erklärung der Features wäre gut. Wenn ein Attribut nominal ist, sollte man die möglichen Werte auflisten. Stehen die Werte noch nicht fest oder gibt es zu viele, kann man einige repräsentativen Werte auflisten.
Die Referenz auf Pradhan et al. 2005 ist gut. Allerdings, wenn man die gleichen (Baseline-)Features wie das angegebene Paper verwendet, soll man dies auch so erwähnen.

siehe Support Vector Learning for Semantic Argument Classication [2] **Classification "siehe [2]" reicht.**

3.3 Evaluation **zur Evaluation siehe später**

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* [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.

Die Formulierung "Wir sind auf die Ergebnisse gespannt" ist ein bisschen zu persönlich, aber es freut mich, wenn es so ist.
Das Ziel, dass ihr im zweiten Teil des zitierten Satzes formuliert, ist ein sehr gutes.

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.

zum Pseudocode:
- Die Methode extFeature() ist schön generell formuliert.
- Die Interpretation von Instanzen ist anhand der Abbildung schwer. Ist ein pbInstance ein Verb, das seinen Argumentrahmen hat? Wie sieht eine Instanz (d.h. eine Zeile) in der ARFF-Datei aus? Vermutung: Jedes Verb-Argument-Paar ist eine Instanz.
- Frage dazu: Wie werden die Argumente extrahiert? Werden die goldenen Annotationen übernommen oder die Argumente heutistisch ermittelt?
- Die innerste for-Schleife ist verwirrend: Soll hier wirklich featureList gefüllt werden? Vermutung: features sollte die Rückgabewerte des jeweiligen feature enthalten.
- Die letze Zeile sollte evtl. zwei Ebenen eingerückt werden, denn soweit ich es verstehe sollte außerhalb der zweiten for-Schleife auf die Liste features nicht mehr zugegriffen werden.

Zur Evaluierung:
Die Evaluierungsideen sind gut.
Wie schon bei den Features erwähnt, wäre es sinnvoll zu erwähnen, dass [2] mit SVM gearbeitet haben - in diesem Sinne baut ihr zuerst mal dieses Werk nach, und zusätzlich kontrastiert ihr die Ergebnisse von [2] bzw. von eurem "Nachbau" mit anderen ML-Verfahren. Diese Vorgehensweise ist eine sehr gute Idee.
Da [2] auch Evaluierungsergebnisse angeben, sollen allerdings diese auch hier schon angegeben werden.

Zu den Belegen:
Wenn man mit der Nummerierung der Literatureinträge im Text arbeitet wie "[2]", dann ist es nicht üblich, jedes Mal auch die Autoren oder den Titel hinzuschreiben. Ihr habt durch Erwähnungen wie "siehe Support Vector Learning for Semantic Argument Classication [2]" eine Mischung aus mehreren Zitations-Traditionen geschaffen:
- "siehe Pradhan et al. (2005)" oder
- "siehe [2]"
(Eine Angabe des Titels ist nicht üblich.)