

Abschlussbericht Semantic Argument Classification

Julian Baumann, Kevin Decker, Maximilian Müller-Eberstein

24. Februar 2015

1 Einleitung

Unter Semantic Argument Classification versteht man, einem Satz seine semantische Rollen wie zum Beispiel Subjekt, Prädikat und Objekt zuzuweisen. Vereinfacht gesagt versucht man zu klären, "wer wem was antut". Sie gehört zum grundlegenden Pre-Processing für verschiedene Natural Language Processing-Anwendungen wie zum Beispiel Machine Translation und ist aus diesem Grunde auch ein wichtiger Aspekt der Forschung. Angelehnt an die Veröffentlichung von [4] versuchten wir im Verlauf unseres Projekts diese Aufgabe automatisch und überwacht durch Machine Learning-Ansätze umzusetzen.

2 Grundlagen

2.1 Daten und Tools

Als Grundlage für unsere Experimente wurden verschiedene Datensätze und Tools verwendet. Wichtigste Ressourcen waren dabei PropBank und Penn TreeBank. PropBank [3] oder auch Proposition Bank ist ein Korpus, in dem für jedes in ihm enthaltene Verb annotiert wurde, welche semantische Rollen es besetzt. Dies ist in Form von sogenannten Frames organisiert, welche auf mehr oder weniger lesbare Weise kodieren, welche Argumente überhaupt gefordert werden können. Im Gegensatz zu FrameNet, welches das gleiche versucht, ist PropBank jedoch nicht ganz so spezifisch sondern versucht durch generellere Argumente für möglichst viele Parser verwendbar zu sein.

Die Penn TreeBank [2] ist ein Subkorpus des Wall Street Journals (WSJ) bestehend aus etwa einer Million Tokens. Davon sind etwa 112.917 Prädikat-Argument Strukturen nach dem PropBank-Schema annotiert. Die oben genannten Korpora bilden die Grundlage für die im Folgenden näher erklärten Experimente. Zur Extraktion der Features wurde die Programmiersprache Python in der Version 3.4 zusammen mit dem Natural Language Toolkit *Version 3.0* verwendet. Die eigentliche Anwendung der Machine-Learning Algorithmen wurde in Weka [1] realisiert.

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

Tabelle 1: Argumente der PropBank

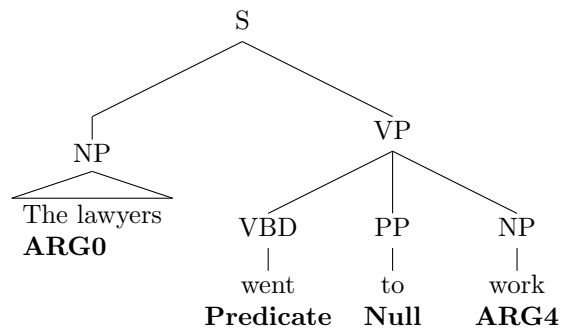
Zur Klassifikation sollten drei verschiedenen Algorithmen verwendet werden: Der Naïve Bayes und der J48-Algorithmus, sowie eine Support Vector Machine. Letztere wurde ausgewählt, um den Experimentausgang mit den Ergebnissen von [4] zu vergleichen, die ebenfalls eine Support Vector Machine verwenden. Wie später im Abschnitt Experimente genauer beschrieben wird, lieferte der Naïve Bayes die zuverlässigsten Ergebnisse.

3 Umsetzung

4 Features

4.1 einzelne Features

Für das Experiment wurden 5 Features extrahiert und ihr Einfluss auf die Ergebnisse getestet. Das erste Feature, **Predicate**, ist einfach die lemmatisierte Form des Prädikats. In unserem Datenset trat dieses Feature in 3966 verschiedenen Werten auf. Zweites Feature, **Path**, beschreibt den Weg, der in einem Syntaxbaum durchlaufen werden muss, um von einem Argument zum Prädikat der Instanz zu gelangen. Dies wird durch \uparrow und \downarrow gekennzeichnet.



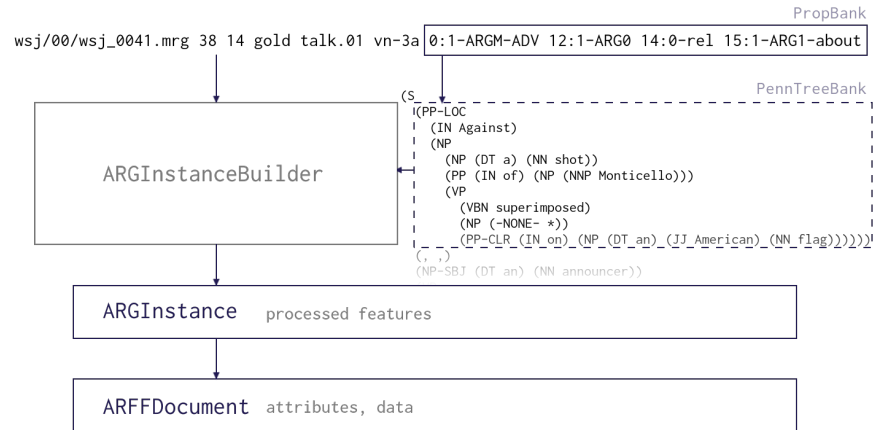
Im Satz „The lawyers went to work“ würde dieses Feature beispielsweise den Wert $\text{NP} \uparrow \text{S} \downarrow \text{VP} \downarrow \text{VBD}$ annehmen. Dieses Feature hat sehr viele verschiedene Ausprägungen(41737 distinct values), die durch verschiedene Satzstrukturen verursacht werden. In der Evaluation betrachten wir ebenfalls Vor- und Nachteile der Vereinfachung dieses Features in der Form von bsp. Duplikatbeseitigung

(... \uparrow S \downarrow VP \downarrow VP \downarrow ... \Rightarrow ... \uparrow S \downarrow VP \downarrow ...) oder Pruning (NP \uparrow S \downarrow VP \downarrow ... \Rightarrow NP \uparrow). Im Feature **Phrase Type** ist kodiert, welchem Konstituententypus (also NP, VP, etc.) das gesuchte Argument angehört. In unserem Fall sind dies 65 verschiedene Kategorien, nachdem für unsere Zwecke zu eingrenzende Typen konvertiert wurden (z.B. NP-SUBJ zu NP). Die anderen beiden Features **Position** und **Voice** sollten beide intuitiv betrachtet binär sein, durch die unvollständige Annotation in Propbank ist dies im Falle von Voice, also ob das Argument passiv oder aktiv ist, nicht der Fall. Hier gibt es noch einen dritten Wert unknown, wenn für das Prädikat keine Angabe diesbezüglich gemacht wurde. Position hingegen ist ein rein binäres Feature und gibt an, ob das Argument vor oder nach dem Prädikat auftritt.

4.2 Featureextraktion

Das Ziel unserer Featureextraktion bestand darin, die Daten der PropBank und Penn TreeBank mit den oben genannten Features in normalisierter Form zusammenzuführen und in das Weka-kompatible Attribute-Relation File Format (ARFF) zu konvertieren.

Instanzen der PropBank liegen in einem Space- und Newline-separierten Textformat vor. Innerhalb einer Instanz verweisen Pointer auf Teilbäume der korrespondierenden Penn TreeBank Sätze, die in einem Klammer-Format annotiert sind.



Mit Hilfe des NLTK, welches die rohen Textdateien beider Korpora in vordefinierte Klassen lädt, erstellen wir ARGInstances, in denen alle Features in normalisierter- und leicht abrufbarer Form vorliegen. Die Helferklasse ARGInstanceBuilder übernimmt die Zusammenführung beider Korpora, die Berechnung des Path-Features mittels Baumtraversierung und Ermittlung des Lowest-Common-Ancestors und ähnliche Extraktionsaufgaben, sowie die finale Normalisierung. Die so generierten ARGInstances werden zuletzt an ein ARFF-Dokument weitergegeben, welches Trainings-, Entwicklungs- und Testdaten in ARFF-Dateien schreibt, die direkt mit Weka weiterverarbeitet werden können.

5 Experimente

5.1 Probleme

6 Evaluation

6.1 Probleme

7 Retrospektive

Literatur

- [1] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1):10–18, 2009.
- [2] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *COMPUTATIONAL LINGUISTICS*, 19(2):313–330, 1993.
- [3] M. Palmer, D. Gildea, and P. Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Comput. Linguist.*, 31(1):71–106, Mar. 2005.
- [4] S. Pradhan, K. Hacioglu, V. Krugler, W. Ward, J. H. Martin, and D. Jurafsky. Support vector learning for semantic argument classification, 2005.