Universität Potsdam
Computerlinguistische Techniken
Dozent:
Prof. Dr. David Schlangen
Wintersemester 2020/21

Bericht

Terminologie Extraktion

Name: Katja Konermann
Martikelnummer: 802658
Email: katja.konermann@uni-potsdam.de

Inhaltsverzeichnis

1 Korpus und Kandidatenauswahl

Der *acl* Korpus besteht aus etwa 11 000 Dokumenten, die insgesamt mehr als 1.5 Millionen Sätzen und 42 Millionen Token enthalten. Die Types belaufen sich auf zirka 520 000 (siehe *Tabelle 1*)
Die Zahl der ausgewählten Bigram-Kandidaten beträgt etwa 340 000.
Zur Auswahl der Kandidaten wurden verschiedene Parameter genutzt. So werden alle Bigramme herausgefiltert, in denen eines oder beider der Token in einer Menge von Stoppwörtern vorkommt. Ich habe dafür die englischen Stoppwörter von *nltk* verwendet. Im Projekt sind sie in der Datei *data/stops_en.txt* zu finden.
Weil Fachbegriffe zumeist aus Nomen bestehen sollten, habe ich zudem Tagging verwendet. Dabei muss ein Bigramm aus zumindest einem relevanten Tag bestehen, um als Kandidat in Frage zu kommen. Mithilfe von *nltk*s *Averaged Perceptron Tagger* werden so alle Bigramme getaggt und die Bigramme, die keinerlei relevante Tags enthalten, herausgefiltert. Als relevante Tags habe ich hier *NN* (Noun, singular or mass), *NNS* (Noun, plural) und *NNP* (Proper noun, singular) genutzt.
Da das *acl* Korpus durch *optical character recognition* erstellt wurde, sind viele Zeichen vorhanden, die keinerlei Bedeutung haben. Um diese Bigramme herauszufiltern, wird bei der Kandidatenauswahl zusätzlich getestet, ob ein Bigramm aus Token besteht, die auschließlich alphabetisch sind.
Die Anzahl der Kandidaten kann außerdem reduziert oder erhöht werden, indem ein Minimum für die absolute Häufigkeit eines Bigramms festgelegt wird. Ich habe für die Kandidatenauswahl hier eine Häufigkeit von 3 gewählt.
Die Kandidaten, die schließlich für die Terminologieextraktion genutzt wurde, sind in der Datei *data/candidates.txt* gespeichert. Um diese Liste von Termen zu reproduzieren, können folgende Argumente bei der Ausführung von *main.py* an die Kommandozeile übergegeben werden:

```
candidates --stops data/stops_en.txt --min_count 3 acl_texts/ <file> NN NNS NNP
```

Dabei sollte <*file*> durch den gewünschten Namen der Ausgabedatei ersetzt werden. Für genauere Information zu den einzelnen Argumenten siehe Abschnitt *Anwendung* oder die *README*.

Dokumente	10 922
Sätze	1 575 233
Token	42 482 606
Types	520 446
Kandidaten (Bigramme)	341 517

Tabelle 1: Korpus und Kandidaten

2 Auswertung

Bei der Extraktion der Fachbegriffe habe ich verschiedene Parameter getestet. Die Werte für den Parameter α habe ich so gewählt, dass verschiedene Gewichtungen der Domänenrelevanz und des Domänenkonsens betrachtet werden können. Der Wert für den Parameter θ wurde dabei für jedes α so gewählt, dass die Anzahl der Kandidaten ungefähr vergleichbar ist, wobei einige Werte für θ absichtlich höher (*output2.csv*) oder niedriger (*output1.csv*) gesetzt wurden, um zu betrachten, wie dies die Leistung des Systems beeinflusst. Die Werte der Parameter mit der dazugehörigen Anzahl der Fachbegriffe sind in *Tabelle 2* aufgeführt.

Für die verschiedenen Parameterwerte werden *Recall*, *Precision* und das harmonische Mittel aus beiden, der *F1-Score*, in *Abbildung 1* dargestellt.

Datei	α	θ	Fachbegriffe
output/output1.csv	0.5	2	20 404
output/output2.csv	0.75	1.75	4 957
output/output3.csv	0.25	2.75	13 495
output/output4.csv	0.9	1.25	9 965
output/output5.csv	0.1	3	15 818

Tabelle 2: Anzahl der Fachbegriffe

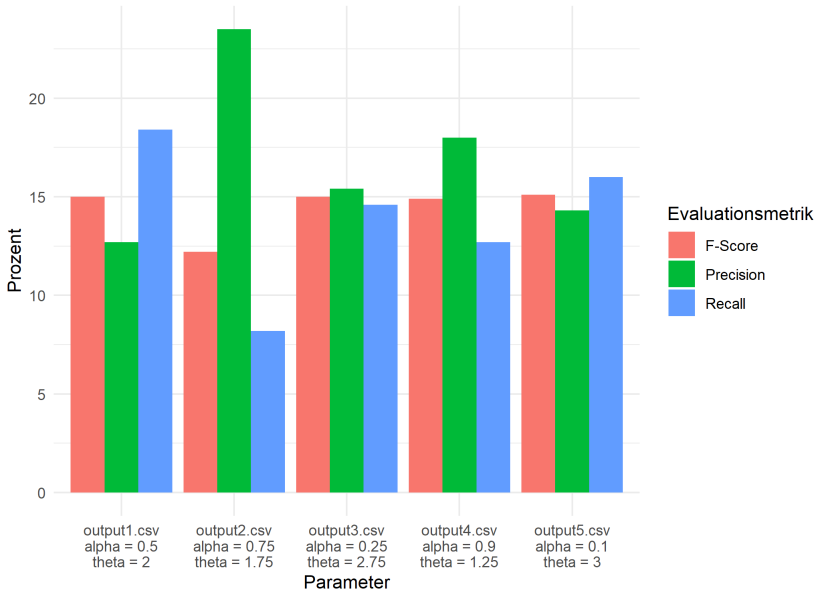

Abbildung 1: Accuracy verschiedener Parameter

3 Bewertung

Mit einer Accuracy von etwa 15% im Durchschnitt weist das Programm eine eher schlechte Leistung auf. Dafür könnte es verschiedene Gründe geben. Zum einen könnte die Vorverabeitung und damit die Kandidatenauswahl verbessert werden. Wie im vorherigen Abschnitt beobachtet enthalten die extrahierten Terme oft noch Bigramme, die etwa aus Verben bestehen. Ich glaube, hier könnte es hilfreich sein, das Tagging noch weiter zu verfeinern, indem beispielsweise eine Tabu-Liste von Tags geschaffen wird, die in den Kandidaten nicht vorkommen dürfen. Als andere Möglichkeit könnte festgelegt werden, dass etwa das zweite Wort in einem Bigram als Nomen getaggt wird, um als Kandidat in Betracht gezogen zu werden.
Ein weiterer Faktor für die schlechte Leistung könnte meiner Meinung nach der Referenzkorpus sein: Es wäre interessant zu untersuchen, ob das System mit einem Korpus, der Paper aus anderen

Forschungsbereichen enthält, eine bessere Leistung schaffen würde. Damit könnten beispielsweise Begriffe wie *et al* oder *paper describes* eher als irrelevant für die Domäne Computerlinguistik eingestuft werden, weil diese auch in anderen Themenbereichen häufig auftreten sollten.
Ein weiteres Problem für das Programm sind zudem die unterschiedlichen Wertebereiche von Domänenkonsens und Domänenrelevanz. Während der Wert eines Begriffs für die Relevanz maximal eins sein kann, ist der Limit für den Domänenkonsens abhängig von der Anzahl der Dokumente. Der maximale Wert der Domänenkonsens ist dabei $ln(N)$, wobei N die Anzahl der Dokumente ist. In diesem Fall heißt das, dass ein Begriff einen Wert von etwa 9.3 erhalten kann, wenn er in allen Dokumenten die gleiche Frequenz aufweist.
Damit fällt der Domänenkonsens oft viel mehr ins Gewicht als die Domänenrelevanz. Um dieses Problem zu umgehen, könnte beispielsweise eine normalisierte Entropie genutzt werden, wobei der vorherige Entropiewert H durch den maximalen Wert $H_{max} = ln(N)$ dividiert wird:

$$H_{norm}(t) = \frac{H(t)}{H_{max}}$$

Damit hätten sowohl Domänenrelevanz als auch -konsens den gleichen Wertebereich $[0, 1]$ und wären so besser vergleichbar.

4 Anwendung

5 Literatur

- https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html