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.

Dabei fällt auf, dass es zwischen den unterschiedlichen Werten für α und θ keine großen Unter-

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

schiede in der Leistung zu geben scheint: Der *F1-Score* pendelt sich oft bei etwa 15% ein. Für $\alpha = 0.5$ und $\theta = 2$ in der Datei *output/output1.csv*, für die die Kandidatenzahl relativ groß ist, ist auch der *Recall*-Wert mit über 18% höher im Vergleich mit den anderen Parametern. Dafür fällt *Precision* mit etwa 13% jedoch im Gegensatz niedriger aus. Umgekehrt ist die Trefferquote von ca. 8% für $\alpha = 0.75$ und $\theta = 1.75$ mit einer Kandidatenzahl von um die 5000 deutlich niedriger, während die *Precision* mit über 20% vergleichsweise hoch ist. Das könnte darauf hindeuten, dass der Anteil von korrekten Termen höher ist, wenn auch die Punktzahl höher ist. Im oberen Wertebereich der extrahierten Terme sind somit eher richtige Fachbegriffe.

Es ist zudem auffallend, dass die *Precision* höher ist, wenn die Domänenrelevanz höher ge-

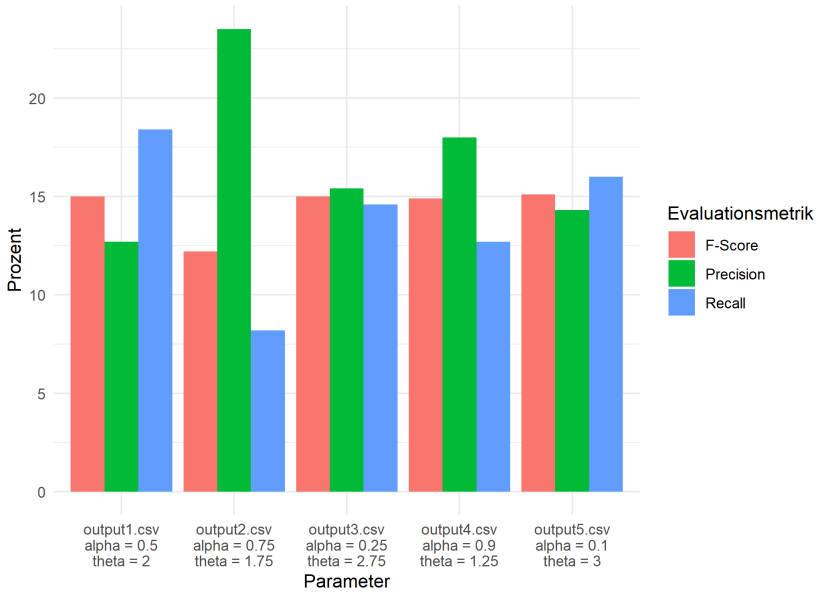

Abbildung 1: Accuracy verschiedener Parameter

wichtet wird, also für Werte von $\alpha > 0.5$. Die *Recall*-Werte sind dagegen höher, wenn dem Domänenkonsens mit Werten für $\alpha < 0.5$ mehr Gewicht zugesprochen wird. Da hier jedoch nur fünf Kombinationen für α und θ ausprobiert wurden, kann nicht eindeutig beurteilt werden, ob dies eine allgemeingültige Tendenz ist. Außerdem ist die Anzahl der extrahierten Terme, für $\alpha > 0.5$ gleichzeitig immer niedriger, was auch die *Precision* erhöhen könnte.

$\alpha = 0.5, \theta = 2$	$\alpha = 0.75, \theta = 1.75$	$\alpha = 0.25, \theta = 2.75$	$\alpha = 0.9, \theta = 1.25$	$\alpha = 0.1, \theta = 3$
et al (4.68)	et al (2.84)	et al (6.53)	et al (1.73)	et al (7.63)
natural language (4.53)	natural language (2.77)	natural language (6.3)	natural language (1.71)	natural language (7.36)
language processing (4.35)	language processing (2.67)	language processing (6.02)	language processing (1.67)	language processing (7.03)
training data (4.14)	training data (2.57)	training data (5.71)	training data (1.63)	training data (6.65)
next section (4.14)	next section (2.57)	next section (5.71)	next section (1.63)	next section (6.65)
computational linguistics (4.12)	computational linguistics (2.56)	computational linguistics (5.68)	computational linguistics (1.62)	computational linguistics (6.62)
future work (4.12)	future work (2.56)	future work (5.63)	future work (1.62)	future work (6.61)
machine translation (4.1)	machine translation (2.55)	machine translation (5.65)	machine translation (1.62)	machine translation (6.58)
test set (4.03)	test set (2.51)	large number (5.58)	test set (1.61)	large number (6.54)
paper describes (4.02)	paper describes (2.51)	test set (5.54)	paper describes (1.60)	total number (6.48)

Tabelle 3: Extrahierte Terme mit der höchsten Punktzahl

$\alpha = 0.5, \theta = 2$	$\alpha = 0.75, \theta = 1.75$	$\alpha = 0.25, \theta = 2.75$	$\alpha = 0.9, \theta = 1.25$	$\alpha = 0.1, \theta = 3$
tagging tool (2)	use simple (1.75)	extraction applications (2.75)	adjacent word (1.25)	intended sense (3)
recognition method (2)	considerable effort (1.75)	equal length (2.75)	constraints expressed (1.25)	new classes (3)
current implementations (2)	many approaches(1.75)	google search (2.75)	discourse features (1.25)	first parser (3)
h c (2)	closer examination (1.75)	specialized domain (2.75)	demonstrative pronoun (1.25)	descriptive adequacy (3)
cornell university (2)	average recall (1.75)	different modalities (2.75)	distinctive feature (1.25)	english preposition (3)
internal semantic (2)	shallow semantic (1.75)	n r (2.75)	best features (1.25)	manual translation (3)
linguistic constituents (2)	language users (1.75)	feature bundles (2.75)	standard machine (1.25)	grained analysis (3)
common terms (2)	training methods (1.75)	translation equivalent (2.75)	multimodal dialogue (1.25)	internet search (3)
subordinate conjunction (2)	integrated approach (1.75)	structural rules (2.75)	pos features (1.25)	intervening material (3)
asr systems(2)	semantic rule (1.75)	phonemic transcription (2.75)	unification process (1.25)	lexical processing (3)

Tabelle 4: Extrahierte Terme mit der niedrigsten Punktzahl

In *Tabelle 3* sind für jede der fünf Parameterkombination die zehn Begriffe aufgeführt, die die höchste Punktzahl erzielt haben. Diese Terme sind über die Kombinationen hinweg meist dieselben. Es scheint hier nur wenig Variation zu geben. Das wird höchstwahrscheinlich daran liegen, dass diese Begriffe über die Dokumente hinweg sehr häufig auftreten und damit einen hohen Konsenswert besitzen. Der Domänenkonsens kann im Gegensatz zur Domänenrelevanz größer als eins sein; sein Maximum ist dabei durch die Anzahl der Dokumente bestimmt (siehe Abschnitt *Bewertung*). Der Einfluss des Domänenkonsens ist damit oft größer als der der Domänenrelevanz. Begriffe, die in allen Dokumenten ähnlich frequent auftreten, werden bei der Extraktion so übermäßig bevorzugt. Darum ist es nicht sehr verwunderlich, dass *et al* für alle Parameter der Terminus mit der höchsten Punktzahl ist. Auch Terme wie *paper describes*, *future works* oder *next section*, die in wissenschaftlichen Arbeiten allgemein häufig auftreten, fallen in diese Kategorie.
In den höchst bewerteten Begriffen finden sich jedoch auch Termini wie *computational linguistics*, *language processing* oder *machine translation*, die durchaus als gute Fachbegriffe angesehen werden können.
In *Tabelle 4* werden für jede der fünf Parameterkombination die zehn Begriffe aufgeführt, die noch als Terminologie angesehen werden, also jeweils θ überschritten, aber die niedrigste Punktzahl erreicht haben. Die Termini sind hier durchmischter, als in den zehn höchst bewerteten Begriffen,

$\alpha = 0.5, \theta = 2$	$\alpha = 0.75, \theta = 1.75$	$\alpha = 0.25, \theta = 2.75$	$\alpha = 0.9, \theta = 1.25$	$\alpha = 0.1, \theta = 3$
heating oil $(6 * 10^{-4})$	heating oil $(9 * 10^{-4})$	heating oil $(3 * 10^{-4})$	heating oil $(1.1 * 10^{-3})$	heating oil $(1.2 * 10^{-4})$
net earnings $(5.8 * 10^{-4})$	net earnings $(8.7 * 10^{-4})$	net earnings $(2.9 * 10^{-4})$	net earnings$(1 * 10^{-3})$	net earnings $(1.2 * 10^{-4})$
net profits $(5.8 * 10^{-4})$	net profits $(8.6 * 10^{-4})$	net profits $(2.9 * 10^{-4})$	net profits $(1 * 10^{-3})$	net profits $(1.2 * 10^{-4})$
mercantile exchange $(5.2 * 10^{-4})$	mercantile exchange $(7.8 * 10^{-4})$	mercantile exchange $(2.6 * 10^{-4})$	mercantile exchange $(9.4 * 10^{-4})$	mercantile exchange $(1 * 10^{-4})$
world oil $(3 * 10^{-4})$	world oil $(4.6 * 10^{-4})$	world oil $(1.5 * 10^{-4})$	world oil $(5.5 * 10^{-4})$	world oil $(6.2 * 10^{-5})$
company spokesman $(2.5 * 10^{-4})$	company spokesman $(3.8 * 10^{-4})$	company spokesman $(1.2 * 10^{-4})$	company spokesman $(4.5 * 10^{-4})$	company spokesman $(5 * 10^{-5})$
monetary policy $(2.3 * 10^{-4})$	monetary policy $(3.4 * 10^{-4})$	monetary policy $(1.1 * 10^{-4})$	monetary policy $(4.1 * 10^{-4})$	monetary policy $(4.5 * 10^{-5})$
prior year $(1.5 * 10^{-4})$	prior year$(2.2 * 10^{-4})$	prior year $(7.3 * 10^{-5})$	prior year $(2.6 * 10^{-4})$	prior year $(2.9 * 10^{-5})$
tender offer $(7.7 * 10^{-5})$	tender offer $(1.1 * 10^{-4})$	tender offer $(3.8 * 10^{-5})$	tender offer $(1.4 * 10^{-4})$	tender offer $(1.5 * 10^{-5})$
central bank $(3.3 * 10^{-5})$	central bank $(4.9 * 10^{-5})$	central bank $(1.6 * 10^{-5})$	central bank $(5.9 * 10^{-5})$	central bank $(6.6 * 10^{-6})$

Tabelle 5: Ausgeschlossene Terme mit der niedrigsten Punktzahl

offensichtlich auch, weil die einzelnen Parameter unterschiedlich viele Begriffe extrahieren. Auch in den am niedrigsten bewerteten Begriffen sind durchaus plausible Termini wie *tagging tool*, *discourse features* oder *lexical processing*. Doch auch hier lassen sich Begriffe erkennen, die eher

bedeutungslos für die Domäne Computerlinguistik zu sein scheinen. Vor allem Bigramme, die zum Teil aus Verben bestehen wie *use simple* und *constraints expressed*, scheinen meiner Meinung nach eher schlecht als Fachbegriffe geeignet zu sein. Das fällt auch schon in den am besten bewerteten Begriffen auf. Außerdem sind unter den am Begriffen auch Worte, die nur aus zwei einzelnen Buchstaben bestehen (*h c*, *n r*). Hier ist mir nicht ganz klar, ob diese für bestimmte Abkürzungen stehen oder durch Fehler beim *optical character recognition* entstanden sind.
In *Tabelle 5* sind die Terme dargestellt, die insgesamt die niedrigste Punktzahl erhalten haben. Ähnlich wie bei den am besten bewerteten Begriffen ist auch hier keine Variation zu erkennen: Über die Parameter hinweg werden Begriffe wie *central bank*, *net profits* oder *heating oils* als Terminologie ausgeschlossen. Tatsächlich habe ich auch den Eindruck, dass diese Begriffe nicht mit der Domäne der Computerlinguistik in Verbindung stehen. In den ausgeschlossenen Termen mit einer höheren Punktzahl finden sich zwar durchaus noch Begriffe, die als Fachterminologie zählen können, aber zumindest die niedrigsten bewerteten Begriffe scheinen auch tatsächlich nicht nur irrelevant sondern auch falsch für die Computerlinguistik zu sein. Stattdessen gehören sie meiner Meinung nach eher in die Domäne der Wirtschaft.
Die in den Tabellen 3, 4 und 5 dargestellten Begriffe sind nur ein sehr kleiner Ausschnitt der extrahierten Terminologie. Trotzdem kann so ein Einblick in die qualitative Leistung des Programms gegeben werden. Die komplette Termlisten mit den jeweiligen Punktzahlen im Verzeichnis *output* zu finden. Wie die Ergebnisse repliziert werden können, kann im Abschnitt *Code* nachgelesen werden.

3 Bewertung

Qualitativ schneidet das System meiner Meinung nach nicht zu schlecht ab. Zwar sind unter den extrahierten Termini durchaus Ausdrücke wie *next section* oder *paper describes*, die eher bedeutungslos für die Computerlinguistik scheinen. Zumindest sind mir aber in den extrahierten Terme keine Begriffe aufgefallen, die offensichtlich nichts mit der Domäne Computerlinguistik zu tun haben, obwohl Begriffe wie *monetary policy* oder *central bank* durchaus in den Kandidaten vorkamen.
Mit einer Accuracy von etwa 15% im Durchschnitt weist das Programm quantitativ jedoch 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 werden muss, um als Kandidat in Betracht gezogen zu werden. Derzeit muss nur eins der beiden Worte in einem Bigramm ein Nomen. Ich glaube, wenn immer das zweite Wort als Nomen getaggt werden sollte, könnten sinnvollere Kandidaten extrahiert 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 wissenschaftlichen Arbeiten 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 $log(N)$, wobei N die Anzahl der

Dokumente ist. In diesem Fall heißt das, dass ein Begriff einen Wert von etwa 9.3 erhalten könnte, wenn er in allen Dokumenten die gleiche Frequenz aufweist.
Damit fällt der Domänenkonsens jedoch 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 für einen Term t durch den maximalen Wert $H_{max} = log(N)$ 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 zu vergleichen und gewichten. Gerade die Kombination von einem anderen Referenzkorpus und der normalisierten Entropie könnte Begriffe wie *et al* zumindest als höchst bewerteten Begriff ausschließen.

4 Code

4.1 Kandidatenauswahl

In der Datei *preprocess.py* befindet sich die gleichnamige Klasse, die einerseits Korpora vorverarbeitet, andererseits aber auch Kandidaten generieren kann. Für die Tokenisierung und das Händeln von Korpus-Verzeichnissen wird hier NLTK's *PlainTextCorpusReader* verwendet. Die Klasse *Preprocess* zählt etwa das Vorkommen von Bigrammen insgesamt oder in einzelnen Dateien (Methode *bigrams()*) oder gibt die Häufigkeit von übergebenen Bigrammen zurück (Methode *get_frequency()*). Außerdem gibt es mehrere Methoden, die Bigramme nach bestimmten Eigenschaften überprüfen: Die Methode *is_lexical()* kontrolliert, ob ein Bigram nur aus alphabetischen Zeichen besteht. Die Methode *has_relevant_tag* taggt ein Bigram mithilfe von NLTK's *averaged perpceptron tagger* und überprüft, ob es mindestens ein Tag aus einer Menge relevanter Tags (Nomen) besitzt.
Diese Kriterien werden in der Methode *candidates()* zusammengeführt, die für jedes Bigram im übergebenen Korpus überprüft, ob es aus alphabetischen Zeichen besteht und relevante Tags besitzt. Zusätzlich kann der Methode eine Liste von Stoppwörtern und eine minimale absolute Häufigkeit übergeben werden, die ein Bigram besitzen muss, um als Kandidat angesehen zu werden.
In der Methode *write_candidates_file()* können dann Kandidaten in einer Textdatei gespeichert werden.
Um selbst Kandidaten zu generieren, sollte die Datei *main.py* genutzt werden. Um anzuzeigen, dass Kandidaten extrahiert werden sollen, muss als erstes das Argument *candidates* gegeben werden. Danach können folgende Argumente übergeben werden:

```
main.py candidates [--stops <file>] [--min <int>] <domain> <output file> [<tag> [<tag> ...]]
```

Unter dem optionalen Argument *–stops* kann eine Datei mit Stoppwörtern übergeben werden. Diese sollte in jeder Zeile ein Wort enthalten. Wird es nicht genutzt, werden auch keine Stoppwörter benutzt. Das Argument *–min* mit einem Integer wird benutzt, um Bigramme herauszufiltern, die nicht die nötige minimale absoulte Häufigkeit besitzen. Wird das Argument weggelassen, ist der Defaultwert eins. Unter dem Argument *<domain>* sollte der Domänenkorpus übergeben werden, hier also das Verzeichnis *acl_texts*. Das Argument *<output file>* kann ein freigewählter Dateiname sein, indem die generierten Kandidaten gespeichert werden sollen.

Als letztes können eine beliebige Menge von Tags übergeben werden, von denen ein getaggtes Bigram mindestens eins enthalten muss, um als Kandidat angesehen zu werden. Werden keine Tags in der Kommandozeile übergeben, wird bei der Extraktion der Kandidaten auch kein Tagging genutzt.

4.2 Terminologie Extraktion

In der Datei *terminology.py* befindet sich die gleichnamige Klasse, die für die Extraktion der Fachbegriffe zuständig ist. Bei der Instanziierung wird hier ein Domänenkorpus, ein Referenzkorpus und eine Menge von Kandidaten übergeben. Die beiden Korpora werden mithilfe der *Preprocess* Klasse vorverarbeitet, das heißt, es werden die Bigramme gezählt und Großschreibung beseitigt. Dann wird Domänenrelevanz und -konsens berechnet.
Mit den Methoden *weight_candidates* und *extract_terminology()* kann dann Terminologie extrahiert werden. Die Methode *write_csv_file()* schreibt außerdem eine Datei, in der alle Kandidaten enthalten sind. Diese Datei ist folgendermaßen aufgebaut: In den ersten zwei Zeilen befindet sich die Werte für α und θ. Danach besitzt jede Zeile drei Spalten. In der ersten Spalte befindet sich der Kandidat, in der zweiten die Punktzahl, die er erreicht hat und in der dritten die Angabe darüber, ob der Kandidat, Terminologie ist oder nicht, also über θ überschritten wurde. Die Terme in der Ausgabedatei sind aufsteigend sortiert, das heißt, der Kandidat mit der höchsten Punktzahl befindet sich oben in der Datei und der Kandidat mit der niedrigsten Punktzahl ganz unten.
Um selbst Terminologie zu extrahieren, sollte die Datei *main.py* genutzt werden. Ihr muss zunächst das Kommando *extract* übergeben werden, um anzuzeigen, dass Terminologie extrahiert werden soll. Dann können folgende Argumente an die Kommandozeile übergeben werden:

```
main.py extract [-a <alpha>] [-t <theta>] <domain> <candidates> <output file>
```

Unter den optionalen Argumente *-a* und *-t* werden die Werte für α und θ spezifiziert. Werden die weggelassen, so wird *-a* als 0.5 und für *-t* als 2 festgelegt. Unter <*domain*> sollte der Domänenkorpus, also hier *acl_texts* übergeben werden, unter <*candidates*> die von dem Kommando *candidates* generierte Kandidatendatei. Soll die Datei benutzt werden, die auch ich verwendet habe, wäre das hier *data/candidates.txt*. Das Argument <*output file*> stellt den Namen der gewünschten Ausgabedatei dar.

4.3 Evaluation

In der Datei *evaluation.py* befindet sich die gleichnamige Klasse, die bei ihrer Instanziierung eine Menge von Termen erhält, die als Goldstandard angesehen werden, und ein *Dictionary* mit gewichteten extrahierten Termen.
In den Methoden *recall()*, *precision()* und *f1()* können die extrahierten Terme quantitativ evaluiert werden. Mithilfe der Methoden *highest_scored()* und *lowest_scored* können zudem die jeweils n höchst- oder niedrigstbewerteten extrahierten Fachbegriffe ermittelt werden.
Die Klasse besitzt außerdem die Klassenmethode *from_file*, die Goldterme und extrahierte Terme aus Dateien ausliest. Die Goldterme sind dabei in einer einfachen Textdatei enthalten, in der in jeder Zeile einen Fachbegriff enthält, während die extrahierten Terme im gleichen csv-Format wie die Ausgabedatei von *extract* gespeichert sein sollten.
Um eine *csv*-Datei zu evaluieren, muss als erstes das Argument *evaluate* beim Ausführen von *main.py* angegeben werden. Dann folgen folgende Argumente:

```
main.py evaluate --extracted <term file> --gold <gold file> [--high <int>] [--low <int>]
```

Unter *–extracted* wird die *csv*-Datei mit der extrahierten Terminologie, unter *–gold* die Textdatei mit den Goldstandardtermen übergeben.
Die optionalen Argument *–high* und *–low* führen dazu, dass die gewünschte Anzahl von hoch- beziehungsweise niedrigbewerteten Termini ausgegeben werden. Hierbei ist zu beachten, dass ausschließlich Terme ausgegeben werden, die auch als Fachbegriffe angesehen werden (θ überschritten haben). Ausgeschlossene Terme tauchen hier nicht auf.

5 Literatur

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