Automatische Verarbeitung deutschsprachiger Tweets: Eine Fallstudie

1 Einführung

TODO: DIE BESONDERHEITEN DES TWEET PROCESSING

- micro-blogging: max. 140 Zeichen (Schwierigkeit für z.B. Sprachidentifikation, Koreferenzerkennung wenn Referent in vielen verstreuten tweets auftaucht, Parsing)
- Sehr hoher "noise": Alle möglichen Arten von Sonderzeichen (user mark-up, smilies, emoticons etc.)
- sehr niedrige Grammatikalität
- Orthographiekonfusion: "Sprechschrift", ad-hoc Abkürzungen, Buchstaben-Zahl-Wörter (z.B. 2b, b4, r8, sk8er), Prolongationen (guuuuuut), Netzjargon
- Hashtags, Retweets
- ...

2 Zielsetzung

Das Projekt "Diskurse in Social Media" untersucht aus kommunikationswissenschaftlicher Perspektive den Verlauf von politischen Diskursen in drei verschiedenen Social Media: Twitter, Facebook und Blogs. Es wird untersucht, inwieweit sich zwischen diesen drei Medien Unterschiede finden lassen im Hinblick auf

- den zeitlichen Verlauf von Debatten: Wie entwickelt sich das Volumen einer Diskussion?

- die Dialogizität der Diskurse: Wie gehen Teilnehmer auf die Beiträge anderer Teilnehmer ein?
- die Meinungsführerschaft: Sind bestimmte Akteure in den Diskussionen "tonangebend"?
- die Themen: Welche Aspekte des Themenkreises werden diskutiert?
- die Meinung: Welche Stimmungslage kommt in einer Diskussion zum Ausdruck?

Die qualitativ hochwertige Analyse dieser Fragestellungen setzt das sachkundige menschliche Urteil voraus, das heißt: Am Ende der Bemühungen steht eine Inhaltsanalyse durch ausgebildete Analysten. Die Projektpartner aus der Wirtschaftsinformatik und der Computerlinguistik sollen diesen Prozess aber maßgeblich unterstützen, um bei einem relativ umfangreichen Datensatz die menschliche Analyse auf die relevanten Beiträge konzentrieren zu können. Als erstes Arbeitskorpus wurden dazu Daten ausgewählt, die im Jahr 2011 über einen Zeitraum von zwei Wochen hinweg (TODO: stimmt das? - wenn die Angabe 12.12-17.02 im Dateinamen des Korpus den Zeitraum des Textsammelns bedeutet, so dürften es gut 2 Monate sein) das Stichwort 'Wulff' beinhalten, also höchstwahrscheinlich Äußerungen zur Affäre um den seinerzeitigen Bundespräsidenten beinhalten. Im vorliegenden Beitrag beschränken wir uns auf die Twitter-Daten, das sind 119 455 einzelne Tweets, von denen wir zunächst 28 818 als Duplikate idetifiziert haben; somit verbleibt eine Grundmenge von 90 637 zu verarbeitenden Tweets.

Im Projekt sind die Wirtschaftsinformatiker für die Beschaffung der Datensätze und die Konstruktion von Graphstrukturen zuständig, die die Verweise zwischen Diskussionsbeiträgen abbilden. Der Computerlinguistik obliegt die inhaltliche Analyse der einzelnen Beiträge im Hinblick auf behandelte Themen und die Stimmungslage: Es wird eine "Vorsortierung" der Beiträge durchgeführt, um im Idealfall die Aufgabe der Analysten auf eine zügige Durchsicht beschränken zu können. Die Vorsortierung erfolgt anhand dreier Dimensionen:

- Themenklassifikation: Durch unüberwachte Verfahren wird ein Clustering von Beiträgen im Hinblick auf (Unter-) Themen vorgenommen. Hier einige Beispiele aus dem Wulff-Korpus, die das Adressieren verschiedener Themen illustrieren: TODO
- Sentimentklassifikation: Für jeden Einzelbeitrag soll erkannt werden, ob eine positive, negative oder neutrale Haltung ausgedrückt wird.

Korpus-Beispiele: TODO
Zusätzlich soll im Falle von Verweisen auf andere Beiträge festgestellt werden, ob diese zustimmend, kritisch, oder neutral ausfallen. Korpusbeispiele: TODO

- Diskursqualitätklassifikation: Textbeiträge können inhaltlich fundiert sein und das Potenzial haben, eine Diskussion fruchtbar voranzubringen, oder lediglich kurze "Einwürfe" darstellen. Korpusbeispiele: TODO

3 Die Pipeline zur Vorverarbeitung

TODO:

3.1 Sprachidentifikation

Die von den Wirtschaftsinformatikern zusammengestellten Daten enthalten auch Tweets, die nicht in deutscher Sprache abgefasst sind. Um diese zu filtern, wurde mit drei *off-the-shelf* Werkzeugen zur Sprachidentifikation experimentiert. Alle drei Sprachidentifizierer arbeiten auf der Basis von N-Grammen, welche zu je einem Sprachmodell für jede zu erkennende Sprache zusammengefasst sind und anhand deren das jeweilige Werkzeug versucht die Sprache(n) zu erkennen.
In folgender Übersicht sind die wichtigsten Eigenschaften der jeweiligen Werkzeuge zusammengefasst:

TextCat

- Textcat: character (byte) N-grams for language guessing -> N-Gram Frequency Profiles
- Trainingsdatensets: Usenet newsgroups
-
-
-
-
-

langid.py

- als Kommandozeilentool, Python-Modul oder Web service verwendbar, da einzelne Datei mit minimalen Abhängigkeiten (somit auch schnell)
- mitgeliefertes Sprachmodell wurde auf 97 Sprachen aus 5 verschiedenen Domänen trainiert
- Trainingsdatensets: Regierungsdokumente, Software Dokumentationen, Nachrichten, Texte aus Online Enzyklopädien und Internet crawl (TODO: Wie das übersetzen?) (= multi-domain language identification corpus, Lui and Baldwin (2011))
- domänenspezifische Features (z.B. HTML, XML, markdown) beeinflussen Sprachidentifikation nicht
- naiver Bayes Klassifizierer mit multinomialem event model trainiert auf Mischung von N-Grammen ($1 \leq n \leq 4$)
- Integration von domänenspezifischen Informationen durch sog. "LD (=Language Domain) feature selection"
- Tokenizierung und Feature Auswahl werden im Eingabedokument mit einem Schritt erledigt (via Aho-Corasick string matching)
- schlägt TextCat bei 7 verschiedenen Test-Korpora (inkl. Micro-Blogs) hinsichtlich Exaktheit und Geschwindigkeit
- von 569 deutschen Tweets im Wulff-Korpus nur 40 mal nicht als Deutsch erkannt

LangGuess

- LangGuess: Detect language of a text using naive Bayesian filter, 99% over precision for 49 languages
- Trainingsdatensets: Wikipedia
-
-

-
-
-

LangId bzw. LangGuess benutzen zusätzlich einen bayesschen Klassifizierer bzw. einen bayesschen Filter. TextCat hingegen errechnet sogenannte "N-Gramm-Frequenz-Profile" auf dessen Basis dann die eigentliche Sprachidentifikation stattfindet.
(Alle drei Sprachidentifizierer sind auf unterschiedliche Art und Weise implementiert. LangId und TextCat sind Python- bzw. Perl-Implementationen, während LangGuess eine Java-Bibliothek ist.)

Um die Präzision der Sprachidentifikation der o.g. Tools einzuschätzen, wurde folgenderweise vorgegangen: Alle Tweets, die einstimmig von allen drei Programmen, als deutsch bzw. nicht-deutsch interpretiert wurden, sind bewusst außer Betracht gelassen worden, da diese keinen Einfluss auf die Entscheidung für die jeweilige Software-Lösung gehabt hätten. Von den übrig gebliebenen 5962 Tweets wurden 10% (d.h. 596 Tweets) zufällig ausgewählt und die Richtigkeit der geratenen Sprachen manuell überprüft. Die Ergebnisse dieser Evaluierung werden in der Tabelle unten zusammengefasst:

Korrekte Entscheidungen		
LangId	TextCat	LangGuess
92.97%	61.34%	35.5%

Tabelle 1: Evaluierung Sprachidentifikation

3.2 Satzgrenzenerkennung

Die generelle Arbeitsweise eines "sentence splitters" besteht darin, Punkte am Ende eines Wortes dahingehend zu disambiguieren, ob es sich um einen Satzende-Punkt oder den Bestandteil einer Abkürzung (wie z.B. in "Nr. 3"), einer Datumsangabe o.ä. handelt. Bei Tweet-Processing wurden aßerdem noch zusätzliche zum Teil störende Faktoren festgestellt, die eine Anpassung der Modulregeln an dieses Textgenre notwendig machten, wie z.B:

- Retweet-Markierungen, die in vielen Fällen in Tweets anstelle eines Punktes Sätze voneinander abgrenzen;
- Missachtung von Groß- und Kleinschreibungsregeln;
- Abgrenzung von Sätzen durch Gedanken- und Schrägstriche.

3.3 Normalisierung und Tokenisierung

TODO: Normalisierung
Für die Tokenisierung wurde ein in Perl geschriebener Tokenizer verwendet, der als Vorverarbeitungstool zusammen mit dem Tree-Tagger erhältlich ist.

3.3.1 Zu behandelnde Phänomene

TODO: Klassifizierte Liste von Dingen, die man behandeln muss

3.3.2 Vorgehen

3.4 Part-of-speech Tagging

Für die automatische Wortartenbestimmung in den Tweets wurden der TNT-Tagger[1] und der Tree-Tagger[2] zur Evaluation herangezogen. Beide Tagger sind probabilistische Part-of-speech-Tagger, jedoch unterscheiden sie sich in ihrer genauen Funktionsweise.
Die allgemeine Funktionsweise von probabilistischen Taggern besteht darin Folgewahrscheinlichkeiten von Wortarten anhand eines manuell annotierten, einsprachigen Korpus zu schätzen (Training) und auf Basis dieses sog. Sprachmodells Wortarten in Texten automatisch zu bestimmen (Tagging). Die Trefferquote eines probabilistischen POS-Taggers hängt im allgemeinen stark von der Qualität und Quantität des annotierten Korpus und des darauf errechneten Sprachmodells ab. Eine weitere Grundlage für das erfolgreiche Tagging ist ein sog. Tagset, welches genau vorgibt zu welcher Wortart ein bestimmtes Wort gehört. Für die Untersuchung deutschsprachiger Tweets wurde von beiden Taggern das STTS-Tagset benutzt[3].
Der TNT-Tagger (Kurzform von "Trigrams'n'Tags") ist ein sehr effizienter POS-Tagger, der in einfacher Weise auf beliebige Sprachen und beliebige Tagsets trainiert werden kann. Im TNT-Tagger ist der Viterbi-Algorithmus

[1] `http://www.coli.uni-saarland.de/~thorsten/tnt/`
[2] `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/`
[3] `http://www.ims.uni-stuttgart.de/projekte/corplex/TagSets/stts-table.html`

für Markov-Modelle zweiter Ordnung implementiert, welcher dafür verwendet wird die wahrscheinlichsten Wortartenfolgen (Uni-, Bi-, und Trigramme) zu berechnen. Die Anzahl der vorkommenden Trigramme reicht aber nicht aus um Folgewahrscheinlichkeiten aller Wortarten verlässlich zu schätzen, was zu unerwünschten Ergebnissen beim Tagging führt. Aus diesem Grund werden zusätzlich Trigrammwahrscheinlichkeiten aus den anderen N-Grammen berechnet. Danach wird aus diesen kontextuellen Frequenzen die höchste Wahrscheinlichkeit für ein bestimmtes Trigramm geschätzt, welches dann in das Sprachmodell aufgenommen wird. Um unbekannte Wörter ebenfalls behandeln zu können, besitzt der TNT-Tagger eine Routine, die anhand von unterschiedlich langen Wortendungen Tags vergibt. Dieses Verfahren beruht auf der Annahme, dass Wortendungen ein starker Indikator für die jeweilige Wortart sind. Diese lexikalischen und die zuvor erwähnten kontextuellen Frequenzen bilden das Sprachmodell des TNT-Taggers. Für das Tagging des Wulff-Korpus wurde das mitgelieferte deutsche Sprachmodell verwendet, welches auf dem NEGRA-corpus[4] trainert wurde. Die Arbeitsweise des Taggers besteht nun im wesentlichen darin, den zu taggenden Wulff-Korpus mit dem trainierten Sprachmodell abzugleichen und daraufhin die POS-Tags zu vergeben.

Die Qualität eines Markov-Modell basierten Taggers hängt maßgeblich von der Größe des Trainingskorpus ab. Je kleiner das Trainingskorpus, desto geringer ist die Trefferquote des Taggers. Ein größeres Korpus erhöht jedoch die Zeit, die für das Training aufgewendet werden muss erheblich. Sehr große Korpora sind außerdem mit einem enormen Arbeitsaufwand für die Erstellung und Annotation verbunden. Zudem ist bei sehr großen Korpora nicht zwangsläufig garantiert, dass sehr seltene Wortartenfolgen später richtig getaggt werden.

Der Tree-Tagger vermeidet diese Probleme indem er zum Schätzen der Folgewahrscheinlichkeiten sog. binäre Entscheidungsbäume benutzt. Diese Entscheidungsbäume werden aus einer Menge von Trigrammen konstruiert und besitzen an ihren Blättern eine Tabelle mit Wahrscheinlichkeiten für bestimmte Wortarten. Diese Methode führt (insbesondere bei kleineren Trainingssets) zu einer höheren Genauigkeit des Taggers. Des Weiteren arbeitet der Tree-Tagger mit diversen eingebauten Lexika (Vollformlexikon, Abkürzungslexikon [, Präfix- u. Suffixlexikon???]), die es ermöglichen Lemma-Informationen auszugeben.

Zusätzlich bietet der Tree-Tagger den Vorteil eines eingebauten Tokenizers, welcher zum tokenisieren des Wulff-Korpus verwendet wurde. Somit entfiel

[4] `http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/`

eine Evaluation eines zusätzlichen Tokenizers. Auf dem Output des Tree-Tagger internen Tokenizers wurden beide Tagger getestet. Für die anschließende Evaluation wurden aus allen Tweets 1000 Sätze zufällig ausgewählt (dies entspricht 19649 Token) und getaggt. Danach wurde die Korrektheit der vergebenen Tags manuell überprüft. Die Ergebnisse waren in ca. 83% der Fälle kongruent, lediglich in ca. 17% (dies entspricht 3338 Token) der Fälle wurden unterschiedliche Tags zugewiesen. Das Ergebnis der Analyse der unterschiedlich getaggten Token ist in unten stehender Tabelle zusammengefasst.

Tagger-Vergleich					
Tree Tagger			TNT Tagger		
von Tree-Tagger gewählter besserer Tag	von TNT-Tagger gewählter schlechterer Tag	Anzahl	von TNT-Tagger gewählter besserer Tag	von Tree-Tagger gewählter schlechterer Tag	Anzahl
NE	NN	312	NE	NN	387
NN	ADJD	122	VVINF	VVFIN	48
NN	VVFIN	81	NN	ADJD	45
NN	ADJA	80	NE	ADJA	37
NN	NE	65	NN	ADJA	35
NE	ADJD	64	NN	NE	32
NE	VVFIN	51	$(	ADJA	31
NE	XY	48	FM	NE	21
VVFIN	VVPP	33	FM	NN	17
NN	CARD	33	PWS	PIS	17
...	...	...	...	...	...
Bessere Zuweisungen:		**1530**	Bessere Zuweisungen:		**1083**
Anzahl gefundener Unterschiede:					**3338** (16.99% aller Token)
Irrelevante Unterschiede:					**725**
Anzahl getesteter Sätze:					**1000**

Tabelle 2: Evaluierung POS-Tagger

4 Inhaltsklassifikation / Auswertung

4.1 Koreferenz

Für eine genauere Analyse des Tweet-Inhalts auf der Satzebene soll eine Koreferenzresolution vorgenommen werden.

TODO:

- Wieviele Pronomen finden wir?
- Wieviele sonstige Korefs?
- Wie ist die Performanz von PoCoRes?
- Performanz auf normalisiertem Input?
- Was sind die typischen Fehler?
- Perspektive: wie weiter? Ist twitter-Koref einfacher als generelle Koref? Was dürfte ein vielversprechendes Verfahren sein?

4.2 "Off-topic"

Die Datenerhebung geschah im Wesentlichen durch keyword matching: Beinhaltet ein tweet das Wort 'Wulff'? Dadurch können gelegentlich tweets in die Suchmenge gelangen, die thematisch nicht relevant sind, weil sie sich auf eine andere Person gleichen Namens beziehen.
TODO: Beispiel
TODO: Kommt das häufig vor?
TODO: Wie gehen wir damit um?

4.3 Subtopiks

4.4 Sentiment

4.5 Diskursqualität

5 Zusammenfassung