Empiri*ST* 2015

Shared Task des *Empirikom*-Netzwerks zur automatischen linguistischen Annotation deutschsprachiger internetbasierter Kommunikation

Richtlinie für die manuelle Tokenisierung von Sprachdaten aus Genres internetbasierter Kommunikation

Michael Beißwenger ▪ Sabine Bartsch ▪ Stefan Evert ▪ Kay-Michael Würzner
unter Mitwirkung von Linda Cedli, Helge Hallfarth, Svenja Metschies und Julian Packheiser
(Stand: 13.09.2015)

1. Worum es geht

Bei der Tokenisierung geht es um die Segmentierung schriftlicher Sprachdaten in Grundeinheiten (z.B. Wörter), auf die anschließend Verfahren für die automatische linguistische Klassifikation und Analyse aufsetzen können – beispielsweise Verfahren zur automatischen Wortartenzuordnung, die jedes Wort-Token einer Wortartenkategorie zuordnen.

Prototypischerweise repräsentiert ein Token eine Wortform und ist im Textverlauf links und rechts durch Leerraum begrenzt wie im folgenden Beispiel:

```
 Petra und Simone gehen ins Kino
␣Petra␣und␣Simone␣gehen␣ins␣Kino␣
```

Bei der Segmentierung eines Textes in Tokens gibt es aber auch Fälle, in denen Tokengrenzen nicht mit Leerraum zusammenfallen. Dies ist beispielsweise der Fall im Falle von Kommata oder Satzschlusszeichen, die jeweils ohne Leerraum an das im Textverlauf vorangehende Wort-Token anschließen. Die Token-Grenze liegt in diesem Fall zwischen der Wortform und dem jeweiligen Interpunktionszeichen. Entsprechend wird bei der Tokenisierung das Interpunktionszeichen vom vorangehenden Wort-Token abgetrennt und als eigenes Token behandelt:

```
 Als ich ihn sah, war es bereits zu spät.
␣Als␣ich␣ihn␣sah␣,␣war␣es␣bereits␣zu␣spät␣.␣
```

Bei der manuellen Tokenisierung wird jedes Token in eine eigene Zeile geschrieben. Leerraum als Trenner zwischen Tokens wird dabei entfernt; Token-Grenzen werden durch Zeilenumbrüche angezeigt:

```
Als
ich
ihn
sah
,
war
es
bereits
zu
spät
.
```

2. Das Format der Ausgangsdaten und Werkzeuge für ihre Bearbeitung

Die Daten, die Sie zur manuellen Tokenisierung erhalten, wurden bereits automatisch in durch Leerraum begrenzte Segmente zerlegt (sog. „White-Space-Tokenisierung"), indem Leerzeichen in Zeilenwechsel umgewandelt wurden. Darüber hinaus wird im IBK-Datenset das Ende eines Postings, im Webkorpora-Datenset das Ende eines Absatzes durch eine Leerzeile (d.h. zwei aufeinanderfolgende Zeilenwechsel) angezeigt.

> Def. *Posting:* ein schriftlicher Nutzerbeitrag in Genres internetbasierter Kommunikation, z.B. ein einzelner Beitrag in einem Chat oder einem Online-Forum, ein Diskussionsbeitrag auf einer Wikipedia-Diskussionsseite, ein Weblog-Kommentar oder ein Tweet. Postings wurden von *einem* Autor verfasst und zu einem bestimmten Zeitpunkt *als Ganze* (en bloc) an den Server übermittelt.

An beliebigen Stellen in den Daten können einzelnen Datensegmenten Metainformationen in Form von XML-Tags beigegeben sein. Im IBK-Datenset werden auf diese Weise z.B. Metainformationen zu Postings, im Webkorpora-Datenset Metainformationen zu einzelnen Artikeln gegeben. XML-Tags müssen bei der Weiterverarbeitung der Datensets als Ganze erhalten werden. Auch wenn sie Spatien enthalten, sind sie als *ein Token* zu behandeln (s. auch Abschnitt 4.7).

Beispiele: XML-Tags mit Metainformationen im IBK-Datenset und im Webkorpora-Datenset:[1]

```
<posting id="chat_01-22" author="TomcatMJ" time="2010-03-
20T14:24:20+01:00"/>
alles
konfetti
bei
euch?

<article id="web01234_01"
url="http://www.linguistik.fau.de/index.shtml"/>
Herzlich
Willkommen
auf
dem
Webauftritt
...
```

In vielen Fällen ist das Ergebnis der White-Space-Tokenisierung nicht zufriedenstellend. Im o.a. Beispiel aus einem Chat-Mitschnitt ist etwa das Fragezeichen nicht vom Wort-Token `euch` ge-

1 Das Format der XML-Tags mit Metainformationen kann in den endgültigen Datensets ggf. anders aussehen als in den hier gegebenen Beispielen.

trennt. Darüber hinaus können verschiedene Phänomene, die typisch für die schriftliche Sprachverwendung in der internetbasierten Kommunikation sind, durch die White-Space-Tokenisierung nicht angemessen behandelt werden.

Ihre Aufgabe ist es, die Ausgangsdaten durch manuelle Nachbearbeitung in eine Form zu überführen, die alle relevanten Einheiten sinnvoll als Tokens repräsentiert. Dabei gehen Sie nach den im Folgenden dargelegten Regeln vor.

Werkzeuge zur Bearbeitung der Daten:

Zur Bearbeitung darf kein Textverarbeitungsprogramm wie Microsoft Word oder Open Office Writer verwendet werden, weil diese Programme proprietäre Formate verwenden, die eine weitere automatische Verarbeitung erschweren, bzw. unerwünschte Zeichenkodierungen und Artefakte erzeugen können. Verwenden Sie stattdessen einen Texteditor. Geeignete, kostenlos verfügbare Programme sind u.a.:

- Microsoft Windows: Notepad++ [http://notepad-plus-plus.org/]
- Mac OS X: TextWrangler [http://www.barebones.com/products/textwrangler/], TextMate 2.0 [http://macromates.com/]
- Linux: gedit, Emacs, Kate (alle mit dem Standard-Paketmanager installierbar)

Die zu tokenisierenden Texte sind im sog. *plain text*-Format (Dateiendung: `.txt`) gespeichert und enthalten keinerlei typographische Hervorhebungen oder Formatierungen. Diesen Zustand gilt es im Sinne der weiteren automatischen Verarbeitung unbedingt zu erhalten, was durch die Verwendung der genannten Texteditoren gewährleistet ist.

3. Status der Richtlinie und Behandlung unklarer Fälle

Die vorliegende Richtlinie wurde in einer Vorversion mit fünf Annotatoren an zwei Standorten (Dortmund und Berlin) am Beispiel von zwei Datensets erprobt. Manuell annotiert wurden dabei jeweils 10.000 Tokens aus dem Dortmunder Chat-Korpus (http://www.chatkorpus.tu-dortmund.de) und 10.000 Tokens aus Wikipedia-Diskussionsseiten. Die Auswertung der Abweichungen und Übereinstimmungen aus diesem Annotationsexperiment sind in die vorliegende Fassung der Richtline ebenso eingeflossen wie Ergebnisse aus einer intellektuellen Analyse von Besonderheiten der schriftlichen Sprachverwendung auf Webseiten (Darmstadt, Erlangen). Trotz dieser Vorarbeiten ist nicht ausgeschlossen, dass bei der systematischen manuellen To-

kenisierung noch Fälle auftreten, die in den nachfolgenden Regeln nicht explizit berücksichtigt sind.

Als generelle Leitlinie für die Behandlung unklarer Fälle gilt:

(1) Suchen Sie für den zu behandelnden Problemfall in der vorliegenden Richtlinie einen möglichst ähnlichen Fall und versuchen Sie, den Problemfall nach Möglichkeit analog zu behandeln.

(2) Sollten Sie in der vorliegenden Richtlinie keinen ähnlichen Fall finden, so ziehen Sie die STTS-Guidelines in der Version von 1999 zu Rate (http://www.sfs.uni-tuebingen.de/resources/stts-1999.pdf). Sie finden dort (neben den Richtlinien zum PoS-Tagging) einige Anmerkungen zur Tokenisierung samt Beispielen insbesondere für Mehrworteinheiten.

(3) Kommentieren Sie die Segmentierungsentscheidung, die Sie nach (1) oder (2) treffen, direkt hinter dem betreffenden Token und machen Sie dabei den Bezug auf diejenige Regel in der vorliegenden Fassung der Richtlinie bzw. in den STTS-Guidelines deutlich, an der Sie sich orientiert haben.

(4) Tritt der Problemfall mehrfach auf, so behandeln Sie alle Fälle konsistent. Geben Sie die unter (3) geforderte Erläuterung im Kommentar zum ersten auftretenden Fall an. Verweisen Sie in allen weiteren Fällen – ebenfalls per Kommentar – unter Angabe der ID des Postings mit dem ersten Fall auf die dazu formulierte Erläuterung. (Zum Einfügen von Kommentaren in die Daten s. Abschnitt 5 dieser Richtlinie.)

4. Die Tokenisierungsregeln nach Phänomengruppen

4.1 Interpunktions- und Sonderzeichen, Binde- und Ergänzungsstriche

Zu den Interpunktionszeichen zählen Komma, Punkt, Doppelpunkt, Semikolon, Ausrufezeichen, Fragezeichen, Gedankenstrich, Anführungszeichen, Schrägstrich sowie runde und eckige Klammern (und auch Spitzklammern, sofern diese nicht Teil eines HTML-Tags sind). Bei der Tokenisierung werden diese Zeichen als Interpunktionszeichen-Tokens behandelt. Drei Punkte als Auslassungszeichen (…) werden dabei nicht in drei einzelne Punkt-Tokens zerlegt, sondern als *ein* Token behandelt:

```
Das
hab
ich
mich
auch
schon
gefragt
...
```

Die Abfolge eines Fragezeichens und eines Ausrufezeichens am Satzende behandeln wir als *ein* Token, da in diesem Fall die beiden Zeichen *zusammen* das Satzschlusszeichen bilden:

```
?!
```

Anführungszeichen und Klammern treten in aller Regel paarweise auf. Bei der Tokenisierung behandeln wir die öffnende und die schließende Klammer bzw. das anführende und abführende Anführungszeichen trotzdem jeweils als einzelne Tokens.

In Fällen, bei denen in substantivischen Wortformen – entgegen der orthographischen Norm – Apostrophenzeichen zwischen Wortstamm und Pluralendung eingefügt werden, behandeln wir das Apostroph als Teil des Tokens:

```
bh's
```

nicht:

```
bh
'
s
```

Gleiches gilt für kontraktierte Formen, bei denen – gemäß § 97 der amtlichen Regelung der deutschen Rechtschreibung – der Apostroph (fakultativ) zur Markierung eines ausgelassenen Lautsegments in der gesprochenen Sprache gesetzt wird:

```
mit'm Fahrrad
```

⇨ segmentiere:

```
mit'm
Fahrrad
```

nicht:

```
mit
'
m
Fahrrad
```

Das gilt auch für Apostrophen am Wortende (Genitivmarkierung bei finalem `<s>` oder Ähnlichem, z.B. `<des Virus'>`).

Sind in einem sprachlichen Ausdruck einzelne Segmente durch Auslassungspunkte oder Sonderzeichen ersetzt, so zählen die Auslassungspunkte bzw. Sonderzeichen als Wortbestandteil:

```
du bist echt ein a...
```

⇨ segmentiere:

```
du
bist
echt
ein
a...  [Auslassungspunkte als Wortbestandteil]
```

```
f*** you!
```

⇨ segmentiere:

```
f***  [Drei Asterisken als Wortbestandteil]
you
!
```

Iterierte Satzschlusszeichen, wie sie gerade in der internetbasierten Kommunikation häufig vorkommen – z.B. mehrfach wiederholte Ausrufe- oder Fragezeichen, fünf oder mehr „Gedankenpünktchen“ anstelle von drei Auslassungspunkten –, werden ebenfalls als *ein* Interpunktions-Token und nicht als eine Abfolge mehrerer Interpunktions-Tokens behandelt:

```
.......
????????
!!!
?!?!?!
```

Eine Ausnahme bilden Fälle in denen ein funktionaler Punkt (Ordinalzahl oder Abkürzung) von einer (langen) Ellipse gefolgt wird:

```
In der 2....
```

⇨ segmentiere:

```
In
der
2.
...
```

Folgen iterierte „Gedankenpünktchen“ und iterierte Satzschlusszeichen ohne trennendes Leerzeichen unmittelbar aufeinander, werden sie als zwei Tokens behandelt:

```
....????!!!!
```

⇨ segmentiere:

```
....
????!!!!
```

Im obigen Beispiel stellt `<????!!!!>` als Ganzes eine iterierte Variante des Satzschlusszeichens `<?!>` dar. Entsprechend wird es als ein Token behandelt und nicht weiter segmentiert.

Bindestriche und Ergänzungsstriche werden nicht als Interpunktions-Tokens behandelt und abgetrennt, sondern als ein Teil des Wort-Tokens behandelt. Der Bindestrich tritt in Zusammensetzungen auf und ist Teil des Kompositums; der Ergänzungsstrich bildet zusammen mit dem Wortteil, den er ausleitet, ein eigenes Token:

```
Laub-
und
Nadelbäume (Ergänzungsstrich)
```

```
Hals-Nasen-Ohren-Arzt (Bindestrich, ein Token)
```

nicht:

```
Hals-
Nasen-
Ohren-
Arzt
```

Sonderzeichen und mathematische Operatoren – `#`, `§`, `$`, `€`, `&`, `%`, `~`, `°`, `+` etc. – werden in gleicher Weise wie Interpunktionszeichen als eigene Tokens behandelt. Das gilt auch dann, wenn Sonderzeichen zusammen mit einer Zahl (und ohne dazwischen eingefügtes Leerzeichen) einen bestimmten Wert-Typ abbilden, z.B. einen Prozentwert oder einen Geldbetrag:

```
10%
```

⇨ segmentiere:

```
10
%
```

```
200€
```

⇨ segmentiere:

```
200
€
```

```
§§48
```

⇨ segmentiere:

```
§§
48
```

```
11+21=32
```

⇨ segmentiere:

```
11
+
21
=
32
```

Steht ein Unterstrich ohne vorangehendes und nachfolgendes Leerzeichen, so wird er als Teil eines größeren Tokens aufgefasst. Dies kommt häufig in Benutzernamen („Nicknames“) vor. Der Nickname wird dann nicht weiter segmentiert, sondern ist als Ganzer ein Eigennamen-Token:

```
Avril_Lavigne
```

Fälle nachträglicher Korrektur, in denen unter Verwendung mathematischer Operatoren angezeigt wird, dass ein oder mehrere Buchstaben in einem vorangehenden Posting zu tilgen oder zu ergänzen sind, werden nicht vollständig zerlegt, sondern sollen als sinnhafte Einheiten erhalten werden: `+s -v`

⇨ segmentiere:

```
+s
-v
```

nicht:

```
+
s
-
v
```

Ist ein mathematischer Operator – ohne eingeschobenes Leerzeichen – aus zwei oder mehr Tastaturzeichen zusammengesetzt, wird er als ein Token behandelt. Steht innerhalb des Operators ein Leerzeichen, die Zeichenfolge ist aber klar als Operator (z.B. als Pfeil) intendiert, soll das (mutmaßlich versehentlich eingefügte) Leerzeichen gelöscht werden:

```
f->d
```

⇨ segmentiere:

```
f
->
d
```

```
f - > d
```

⇨ segmentiere:

```
f
->
d
```

In Postings enthaltene Codesegmente werden ebenfalls als ein Token behandelt und nicht künstlich in ein Interpunktions-Token und ein Wort-Token getrennt:

```
/ig schdöbbs
```

⇨ segmentiere:

```
/ig
schdöbbs
```

Sogenannte Doppeltokens – wie *(Neu-)Veröffentlichung*, das gleichzeitig das Lexem *Veröffentlichung* und das Lexem *Neuveröffentlichung* umfasst – werden als separate Tokens (mit Ergänzungsstrich) behandelt:

```
(Neu-)Veröffentlichung
```

⇨ segmentiere:

```
(
Neu-
)
Veröffentlichung
```

Eine Ausnahme von dieser Regel bilden genderneutrale Gruppenbezeichner wie `<Student(inn)en>`, die als *ein* Token angesehen werden:

```
Student(inn)en
```

⇨ keine Segmentierung

Andere Varianten dieses Phänomens werden analog behandelt (vgl. `<StudentInnen>`, `<Student/innen>`). Koordinationen wie `<Experten/Expertinnen>` sind allerdings immer zu segmentieren:

```
Experten/Expertinnen
```

⇨ segmentiere:

```
Experten
/
Expertinnen
```

Im Falle einer Informationskompression von Lexemen, die geklammert und ohne Leerzeichen geschrieben werden, werden gleichfalls die Klamern und sonstigen Interpunktionszeichen segmentiert:

```
(“normale”/saisonale) Grippe
```

⇨ segementiere:

```
(
“
normale
”
/
saisonale
)
Grippe
```

Bei durch Leerzeichen verbundenen Kompositionsgliedern wird entgegen der linguistischen Plausibilität am Leereichen getrennt:

```
Vitamin C-Mangel
```

⇨ segmentiere:

```
Vitamin
C-Mangel
```

Auch in Fällen von „falschen“ Leerzeichen wird die Trennung am Leerzeichen durchgeführt:

```
Magen- Darm- Erkrankung
```

⇨ segmentiere:

```
Magen-
Darm-
Erkrankung
```

Anführungsstriche bei quotierten Bindestrichkompositionsgliedern sind abzutrennen:

```
Oster-„Beckerei“
```

⇨ segmentiere:

```
Oster-
„
Beckerei
“
```

Dies gilt auch für komplexe Eigennamen der Art `<i-„Pott“>`.

Standardregel für nicht oder schwer entscheidbare Fälle:

In Fällen, in denen ein Interpunktions- oder Sonderzeichen ohne Leerzeichen mit einem sprachlichen Ausdruck verbunden ist, sich aber weder eindeutig als Wortbestandteil, noch eindeutig als Tippfehler, noch (unter Anwendung aller bekannten Regeln) eindeutig als vom Wort zu trennendes Token behandeln lässt, gilt als Standardregel: Das Interpunktions-/Sonderzeichen wird nicht vom sprachlichen Ausdruck abgetrennt, d.h. das Ergebnis der ursprünglichen „White-Space-Tokenisierung“ wird unkorrigiert beibehalten.

4.2 Summen-, Zeit- und Datumsangaben

Zahlen, Zeitangaben und andere numerische Werte, die Dezimalkomma, Punkt, Doppelpunkt und/oder andere Interpunktions- bzw. Sonderzeichen beinhalten, werden ebenfalls nicht getrennt, sondern als ein Token behandelt:

`6.200` (ohne weitere Segmentierung)
`6.200,-` (ohne weitere Segmentierung)

`das dauert 2:40 std.`

⇨ segmentiere:

```
das
dauert
2:40
std.
```

Das gleiche Prinzip wird auf negative bzw. gebrochene Zahlen angewendet:

`-1,5`

⇨ nicht zu segmentieren!

`1/2 h`

⇨ segmentiere:

```
1/2
h
```

Es ist zu beachten, dass mit Leerzeichen ausgeführte Brüche als Division interpretiert und dementsprechend tokenisiert werden:

`1 / 2 h`

⇨ segmentiere:

```
1
/
2
h
```

Angaben zu Wertebereichen (z.B Zeiträumen, Häufigkeiten, Mengen) werden hingegen bei der Tokenisierung in ihre Segmente zerlegt.

`14:00-18:00`

⇨ segmentiere:

```
14:00
-
18:00
```

```
14-18 Uhr
```

⇨ segmentiere:

```
14
-
18
Uhr
```

```
1-3 Semester
```

⇨ segmentiere:

```
1
-
3
Semester
```

```
30-50kg
```

⇨ segmentiere:

```
30
-
50
kg
```

Bei komplexen Maßeinheiten orientiert sich die Tokenisierung an der ausgeschriebenen Form:

```
ca. 20°C
```

⇨ segmentiere:

```
ca.
20
°
C
```

```
ca. 20 GHz
```

⇨ segmentiere:

```
ca.
20
GHz
```

Im folgenden Fall realisiert `<mal>` zusammen mit der Zahl `<11>` das Adverb *elfmal*. Entsprechend wird der Ausdruck `<11mal>` nicht weiter zerlegt (analog dazu `<11x>`):

```
4-11mal
```

⇨ segmentiere:

```
4
-
11mal
```

Datumsangaben werden in ihre Bestandteile zerlegt. Punkte, die anzeigen, dass es sich bei einer Zahl um eine Ordinalzahl handelt, werden dabei als Teil des Zahl-Tokens behandelt und nicht abgetrennt:

```
16.07.2013
```

⇨ segmentiere:

```
16.
07.
2013
```

```
16. Juli 2013
```

⇨ segmentiere:

```
16.
Juli
2013
```

Diese Strategie wird auch auf fremdsprachliche Datumsangaben angewendet, etwa bei `<21/07/1980>`:

```
21/
07/
1980
```

Zeitangaben, die beschreibende Phrasen umfassen, werden wie der restliche Text behandelt, d.h. jede Wortform wird segmentiert:

```
Weimarer Klassik
```

⇨ segmentiere:

```
Weimarer
Klassik
```

Bei Datumsangaben, die römische Ziffern enthalten, wird analog und wie folgt verfahren:

```
Hälfte des XIX Jh. = Anfang 1796
```

⇨ segmentiere:

```
Hälfte
des
XIX
Jh.
=
Anfang
1796
```

Ausdrücke, die aus einem Kurzwort (Akronym) und einer Zahl bestehen, werden in zwei Tokens zerlegt:

`WS04` (für „Wintersemester (20)04“)

⇨ segmentiere:

`WS`
`04`

Mischkomposita aus Zahlen und Buchstaben, die ein Lexem bilden, werden nicht segmentiert:

`24-stündig` (nicht segmentiert)

4.3 Abkürzungen und netztypische Akronyme

Einfache Abkürzungen werden als ein Token behandelt. Punkte, die in einer Abkürzung vorkommen, werden entsprechend nicht abgetrennt, sondern als Teil des Tokens behandelt. Bei mehrgliedrigen Abkürzungen orientiert sich die Tokenisierung an der ausgeschriebenen Form: Handelt es sich um separate Wörter, so wird die mehrgliedrige Abkürzung in einfache Abkürzungen aufgetrennt; Abkürzungen von Bindestrichkomposita bleiben als ein Token erhalten.

`etc.`	⇨	`etc.`
`d.h.`	⇨	`d.`
		`h.`
`d. h.`	⇨	`d.`
		`h.`
`o.ä.`	⇨	`o.`
		`ä.`
`u.dgl.`	⇨	`u.`
		`dgl.`
`u. dgl.`	⇨	`u.`
		`dgl.`
`std.`	⇨	`std.`
`Mat.-Nr.`	⇨	`Mat.-Nr.`

Fällt am Satzende ein Abkürzungspunkt mit einem Satzschlusszeichen-Punkt zusammen (= Doppelfunktion des Punktes), so wird der Punkt als Abkürzungspunkt interpretiert und zusammen mit der Abkürzung als *ein Token* behandelt:

`Wir kauften Socken, Hemden, Schuhe etc.`

⇨ segmentiere:

`[…] etc.`

Wird das Abkürzungszeichen bei mehrteiligen Abkürzungen vergessen (`<zB>`), so sind diese nicht zu trennen.

Kürzungszeichen innerhalb eines Wortes werden als Wortbestandteile behandelt. In diesem Fall wird analog zu den Bindestrichkomposita nicht weiter segmentiert:

`Zeitschr.titel` [keine weitere Zerlegung, ein Token]

Abkürzungen in der Form von Mischkomposita aus Buchstaben und Zahlen werden nicht getrennt, wenn sie ein Lexem bilden und ohne Leerzeichen geschrieben werden. Ein Beispiel sind Namen von Verkehrswegen, wie A9 für Autobahn 9:

`A9`

⇨ segmentiere:

`A9`

Aber: Wenn durch ein Leerzeichen getrennt `A` und `9` auftritt, wird nicht zusammengeführt.

Netztypische Akronyme werden, wenn sie kein Leerzeichen enthalten, auch dann, wenn sie für Mehrwortausdrücke stehen, als ein Token behandelt:

`cu Peter` („cu“ = „see you“)

⇨ segmentiere:

`cu`
`Peter`

`Bruce Springsteen` **`aka`** `The Boss` („aka“ = „also known as“)

⇨ segmentiere:

`Bruce`
`Springsteen`
`aka`
`The`
`Boss`

4.4 Behandlung von Tippfehlern und Schnellschreibphänomenen

Tippfehler in den Ausgangsdaten werden bei der manuellen Nachbearbeitung standardmäßig nicht korrigiert – selbst dann nicht, wenn der Autor eines Postings versehentlich ein Leerzeichen an falscher Stelle gesetzt hat, so dass die White-Space-Tokenisierung die betreffende Wortform in zwei Einzel-Tokens aufgespalten hat. Diese Stellen sind aber mit dem Kommentar „`%% Fehler`“ zu versehen (vgl. Abschnitt 5).

Originaldaten (Chat-Mitschnitt):

```
die stehen da schona ber ohne preise
```

⇨ Ausgangsdaten nach White-Space-Tokenisierung bleiben unverändert, werden aber kommentiert:

```
die
stehen
da
schona      %% Fehler
ber         %% Fehler
ohne
preise
```

In solchen Fällen wird das Ergebnis der White-Space-Tokenisierung nicht manuell korrigiert. Hingegen werden Emoticons und andere aus Interpunktions- bzw. Sonderzeichen zusammengesetzte Einheiten, die klar als solche erkennbar sind, auch dann zu einem Token zusammengefügt, wenn dazwischen Leerzeichen stehen (in Analogie zu mathematischen Operatoren usw.). In Zweifelsfällen ist auch hier die White-Space-Tokenisierung beizubehalten.

Originaldaten (Chat-Mitschnitt):

```
tag quaki : )
```

⇨ Ausgangsdaten nach White-Space-Tokenisierung:

```
tag
quaki
:
)
```

⇨ segmentiere:

```
tag
quaki
:)
```

In Fällen, in denen zwischen zwei Wortformen ein Leerzeichen versehentlich vergessen wurde, gilt analog: Es wird nicht künstlich ein Leerzeichen eingefügt, sondern es wird die Zeichenfolge als *ein* Token behandelt und mit einem Kommentar versehen (“typo” für “Tippfehler”):

```
Ich
hab
da
noch
maldrüber     %% typo
nachgedacht
```

Fehlt hingegen ein Leerzeichen zwischen einem Interpunktionszeichen und dem folgenden oder vorangehenden Wort, wird eine Tokengrenze eingefügt (s. dazu auch das Beispiel in Abschnitt 4.6 zu bewusst ausgelassenen Leerzeichen nach Satzschlusszeichen):

```
Anna,kannst du mal […]
```

⇨ Segmentierung:

```
Anna
,
kannst
du
mal
```

```
handfest un direkt- so sind se...die Pottler
```

⇨ Segmentierung:

```
handfest
un
direkt
-
[…]
```

Tippfehler aller Art – Buchstabendreher, Zeichenauslassungen, versehentlich wiederholte Realisierungen desselben Zeichens, versehentliche Großschreibung usw. –, die zu einer orthographisch irregulären Schreibung führen, bleiben unkorrigiert. Das gilt auch für den folgenden Fall, in welchem das Nicht-Betätigen der Hochstelltaste in der Realisierung des Graphems <ß> anstelle des vermutlich anvisierten Satzschlusszeichens <?> resultiert:

```
Warst du vom Zeugnis überraschtß
```

⇨ Segmentierung:

```
Warst
du
vom
Zeugnis
überraschtß
```

4.5 Kontraktierte Formen

Kontraktierte Formen, die typisch für die gesprochene Sprache sind und bei denen zwei (oder mehrere) Wortformen koartikulationsbedingt zusammengezogen werden, werden wie ein Token behandelt. Das heißt: Bei der manuellen Nachbearbeitung der White-Space-Tokenisierung werden diese Formen nicht künstlich getrennt.

Beispiele für typisch sprechsprachliche kontraktierte Formen sind:

– Präposition + Artikel: *innem, aufm*, *aus(s)n*
– Adverb + Artikel: *noch(e)n*
– Konjunktion + Personalpronomen: *fallste* (< *falls du*) *obse* (< *ob sie*)
– Auxiliarverb + Personalpronomen: *haste, biste*
– Vollverb + Personalpronomen: *machste*, *gehste, denkste, schreibste*
– Vollverb + zwei Personalpronomina: *machstes*, *gibstes*
– Kopulaverb + Personalpronomen: *warens*
– Modalverb + Personalpronomen: *kannste, willste*, *sollste, darfste*
– Auxiliarverb + Abtönungspartikel: *(was) isn (passiert)*

Bei kontraktierten Formen, die ein Apostroph enthalten, betrachten wir das Apostroph als Teil des Tokens.

`stimmt’s` (keine weitere Segmentierung)
`waren’s` (keine weitere Segmentierung)

4.6 Verwendung von Binnenmajuskeln (CamelCase) als Wortgrenzenmarkierung

In manchen Formen internetbasierter Kommunikation, in denen für Postings nur eine begrenzte Zeichenzahl zur Verfügung steht (160 Zeichen in SMS-Nachrichten, 140 Zeichen in Tweets), wird bisweilen aus Gründen der Platzersparnis auf die Verwendung von Leerzeichen zur Worttrennung verzichtet; stattdessen werden Binnenmajuskeln zur Anzeige von Wortgrenzen verwendet. Wir behandeln diese Art des Einsatzes von Binnenmajuskeln als eine spezielle orthographische Konvention der Wortgrenzenmarkierung. Kennzeichnet eine Binnenmajuskel eine Wortgrenze, so ist bei der Tokenisierung eine künstliche Tokengrenze zu konstituieren. Gleiches gilt, wenn nach einem Satzschlusszeichen aus Ökonomiegründen auf ein Leerzeichen verzichtet wird (vgl. hierzu auch Abschnitt 4.4):

Beispiel:

```
Zu welchemHandlungsbereich gehört unsereKomm hier?
Bildung?Freizeit?Mischung?
```

⇨ Segmentierung:

```
Zu
welchem
Handlungsbereich
gehört
unsere
Komm
hier
?
Bildung
?
Freizeit
?
Mischung
?
```

Da eine klare Abgrenzung zwischen versehentlicher Zusammenschreibung (s. Abschnitt 4.4) und absichtlichem CamelCase in vielen Fällen nicht möglich ist, wird diese Regel auf *alle* Genres angewendet, nicht nur auf Genres, in denen für Postings eine maximale Zeichenzahl vorgegeben ist (z.B. 140 Zeichen für Tweets in Twitter oder 160 Zeichen pro SMS-Nachricht). Das bedeutet insbesondere, dass bei `<derText>` ein vergessenes Leerzeichen ergänzt wird, bei `<maldrüber>` aufgrund des nicht vorhandenen Binneninitials aber nicht. Eigennamen mit CamelCase wie `<PepsiCo>` werden selbstverständlich nicht getrennt.

4.7 HTML/XML-Tags

HTML/XML-Tags, die in den Ausgangsdaten enthalten sind, werden bei der Tokenisierung *nicht zerlegt* – auch dann nicht, wenn es sich um Elemente mit komplexen Attributen handelt (die Leerzeichen enthalten). Tags, egal welchen Komplexitätsgrads, werden als *ein Token* behandelt, das in diesem Fall Leerzeichen enthalten darf.

Beispiel:

```
Auf
der
englischen
Wikipedia
gab
es
bereits
eine
```

```
<A target="_blank" href="https://en.wikipedia.org/w/index.php?
tit-le=Talk:PRISM_(surveillance_program)&oldid=559238329#Known
_Counter_Measures_deleted_.21">  [ ein Token ]
umfangreiche
Diskussion
</A> [ ein Token ]
zu
diesem
Abschnitt
```

Neben HTML/XML-Tags, die bereits in den Ausgangsdaten enthalten waren, enthalten die beiden Datensets regelmäßig XML-Tags, die nachträglich eingefügt wurden, um einzelnen Datensegmenten Metainformationen beizugeben (s. die Erläuterungen und Beispiele in Abschnitt 2). Auch diese XML-Tags müssen bei der Weiterverarbeitung als ein Token erhalten und dürfen nicht weiter segmentiert werden.

4.8 E-Mail-Adressen und URLs

E-Mail-Adressen und URLs werden grundsätzlich nicht segmentiert und als ein Token behandelt. Das gilt sowohl für reguläre wie auch für gekürzte URLs:

```
michael.beisswenger@tu-dortmund.de   (ohne weitere Segmentierung)
https://en.wikipedia.org/wiki/Main_Page   (ohne weitere Segmentierung)
http://www.shortnews.de   (ohne weitere Segmentierung)
shortnews.de   (ohne weitere Segmentierung)
```

Satzzeichen nach URLs (also solche, die eindeutig nicht zur URL gehören) sind abzutrennen.

4.9 Emoticons

Emoticons werden als eigene Tokens behandelt. Sie werden nicht als eine Abfolge mehrerer Interpunktions- oder Sonderzeichen-Tokens dargestellt. Steht zwischen einem Emoticon und dem vorangehenden oder nachfolgenden Wort-Token kein Leerzeichen, so wird die Grenze trotzdem als Tokengrenze interpretiert.

Emoticons mit iterierten Teilen (z.B. mehrfache Wiederholung der schließenden Klammer beim lachenden Smiley) werden wie Emoticons ohne iterierte Teile behandelt.

Beispiele für Emoticons sind:

```
:-)
:-)))))
;-)
:)
;)
```

```
:-(
8)
:D
^^
o.O
oO
\O/
\m/
```

Tippfehler in Emoticons werden – sofern es sich dabei nicht um irregulär eingefügte Leerzeichen handelt – bei der Tokenisierung ignoriert, d.h.: die Zeichenfolge wird trotz des Tippfehlers als *ein* Token behandelt:

```
:;))
_))
```

Ist ein Emoticon durch Leerzeichen unterbrochen, aber eindeutig als Emoticon intendiert, wird das Leerzeichen getilgt und das Emoticon als ein Token rekonstruiert (vgl. hierzu auch Abschnitt 4.4):

Originaldaten (Chat-Mitschnitt):

```
tag quaki : )
```

⇨ Ausgangsdaten nach White-Space-Tokenisierung:

```
tag
quaki
:
)
```

⇨ Manuelle Segmentierung:

```
tag
quaki
:)
```

Fällt ein Emoticon mit einer Klammer zusammen (= Doppelfunktion des Klammerzeichens einmal als Klammer und einmal als Element des Emoticons), so wird bei der Tokenisierung das Emoticon und nicht die Klammer als Token konstituiert (vgl. analog die Regelung für den Zusammenfall von Abkürzungspunkt und Satzschlusszeichen-Punkt, Abschnitt 4.3):

```
Das hab ich auch schon gehört (find ich super :-)
```

⇨ Segmentierung:

```
(
find
ich
super
:-)
```

4.10 Aktionswörter (Inflektive und Inflektivkonstruktionen)

Aktionswörter werden zur (spielerischen) Beschreibung von Gesten, Mimik, mentalen Zuständen oder „virtuellen“ Handlungen verwendet. Typischerweise sind sie nicht syntaktisch integriert, sondern stehen einleitend oder ausleitend zu einem Satz oder Posting oder werden – ähnlich wie Interjektionen – dazwischengeworfen (realisieren also Parenthesen und keine Satzteile). Typischerweise basieren Aktionswörter auf einem unflektierten Verbstamm (einem sog. ‚Inflektiv‘), der entweder alleine steht oder um weitere Einheiten erweitert sein kann – zum Beispiel um vom Verb geforderte Ergänzungen oder um Angaben. Solche komplexen Aktionswörter werden von den Autoren in den allermeisten Fällen zusammengeschrieben; im Einzelfall können sie aber auch getrennt stehen. Sehr verbreitet ist die Markierung von Aktionswörtern mit ein- und ausleitenden Asterisken, bisweilen auch mit Spitzklammern.

Bei der Tokenisierung sollen Aktionswörter als *ein* Token behandelt werden, sofern sie nicht durch Leerzeichen unterbrochen sind. Sind Aktionswörter durch Leerzeichen unterbrochen, werden sie als mehrere Token behandelt.

Sind Aktionswörter durch Asterisken ein- und ausgeleitet, werden diese Sonderzeichen abgetrennt.

Beispiele:

```
*grübel*
```

⇨ Segmentierung:

```
*
grübel
*
```

```
*auf locher rumhüpf & konfetti mach*
```

⇨ Segmentierung:

```
*
auf
locher
rumhüpf
&
konfetti
mach
*
```

In Ausnahmefällen finden sich auch Aktionswörter mit finiter Verbform. Diese werden analog behandelt:

```
*dichmalganzdolleknuddelt*
```

⇨ Segmentierung:

```
*
dichmalganzdolleknuddelt
*
```

```
immer noch nicht fassen kann
```

⇨ Segmentierung:

```
immer
noch
nicht
fassen
kann
```

Tippfehler bei der Markierung von Aktionswörtern mit Asterisken werden bei der Tokenisierung ignoriert. D.h.: Auch wenn nur an einem Ende des Aktionsworts ein Sternchen steht *oder* wenn anstelle eines Sternchens ein anderes Zeichen (z.B. ein Pluszeichen) steht, wird der gesamte Ausdruck nach den beschriebenen Segmentierungsregeln zerlegt:

```
quakirenntdemflöppymitdeneiswürfelnnach*
```

⇨ Segmentierung:

```
quakirenntdemflöppymitdeneiswürfelnnach
*
```

```
+s*
```

(= Kurzform für ‚smile' –, obwohl das einleitende Sternchen – vermutlich versehentlich – durch ein Pluszeichen ersetzt ist)

⇨ Segmentierung:

```
+
s
*
```

Folgen zwei Aktionswörter unmittelbar aufeinander und steht zwischen beiden nur *ein* Asterisk, so wird links und rechts des Asterisken eine Tokengrenze eingefügt:

```
*danachdenraminmeinenrechnereinbau*G*
```

⇨ segmentiere:

```
*
danachdenraminmeinenrechnereinbau
*
G
*
```

Aktions*beiträge* in Chats, mit denen der Chatter Zustände und Handlungen seines Chat-Charakters aus einer fiktiven Außensicht (in der 3. Person) beschreibt, zählen nicht zu den Aktionswörtern. Sie werden als Sätze aufgefasst und entsprechend regulär in Tokens zerlegt:

```
quaki knuddelt Thor
```

⇨ segmentiere:

```
quaki
knuddelt
Thor
```

4.11 Adressierungen

Adressierungen bestehen in der Regel aus einem Adressierungsmarker – häufig `@` oder `2`, bisweilen auch `zu` – und der Angabe eines Adressaten oder einer Adressatengruppe. Sie werden nicht weiter segmentiert, sondern als *ein* Token behandelt:

```
@bine23
@alle
```

Die Zahl `23` in der Adressatenangabe `bine23` im ersten der beiden o. a. Beispiele wird nicht abgetrennt, da es sich in diesem Fall um einen Namensbestandteil (Teil des Nicknames) handelt.

Ist eine Adressierung unmittelbar (ohne Leerzeichen) an einen vorangehenden Ausdruck angehängt, wird eine Tokengrenze eingefügt. Im folgenden Beispiel ist die Adressierung ohne Leerzeichen an ein Aktionswort (ohne Asterisken-Markierung) angehängt. Das Aktionswort wird bei der Segmentierung als eigenständiges Token behandelt:

```
winke@bochum
```

⇨ segmentiere:

```
winke
@bochum
```

Sind Adressierungen einem Posting oder einem Satz vorangestellt, werden sie häufig durch einen Doppelpunkt von der nachfolgenden Äußerung abgetrennt. Der Doppelpunkt wird in diesem Fall ganz normal als Interpunktionszeichen behandelt und abgetrennt, d.h. als eigenständiges Token konstituiert:

```
@bine23
:
hallöchen
!
:-)
```

4.12 Hashtags

Hashtags bestehen aus dem Hashtagmarker <#> und der Angabe eines Themenbezeichners. Sie werden nicht weiter segmentiert, sondern als *ein* Token behandelt:

```
#urlaub
#SPD
```

4.13 Fremdsprachliches Material

Fremdsprachliche (z.B. englische) Postings oder Teile von Postings werden genauso tokenisiert wie deutsche.

In den Webdaten findet sich häufig eine Mischung aus deutsch-sprachigem und fremdsprachlichem Material, besonders englischsprachige Phrasen und Begriffe, aber auch Einzeltokens, die entweder komplett oder teilweise aus einer anderen Sprache entlehnt wurden.

> **The Left Foot of God** aka **Bronislaw Bitchinski** knüpft unter der Verwendung der **Boss RC-50 Loop Station** kontigent generiierte **Groove**parzellen zu einem **Sound**teppich, der endlosen Improvisationen als Standfläche dient. Dafür kann auf andere Musiker verzichtet werden. Die Gage wandert auschließlich in eine Tasche.
>
> Die überlagerten Doppelbesetzungen bei Ornette Coleman und **Prime Time**, der stampfende Gitarrenbeat von Bohannon, und der freie Gitarrengegensatz von **Singlenot-Funkriff** und Brett**funk**-Vermettung bei Defunkt.

Fremsprachliche Tokens (z. B. `Loop`), Phrasen (z. B. `The Left Foot of God`) und Namen (`Bronislaw Bitchinski`) werden wie deutsches Material tokenisiert, ebenso Mischkomposita (z. B. `Soundteppich, Brettfunk`).

4.14 Strukturierte Darstellungsformate: Tabellenartige oder pseudotabellarische Anordnung des Textes

In den Webtexten kommen bisweilen tabellenartig angeordnete Daten vor, die neben Text auch Zahlen und Beträge umfassen können. Diese werden bei der Tokenisierung wie der übrige Fließtext behandelt, d.h. zeilenweise von links nach rechts bearbeitet. Insbesondere soll die Reihenfolge der Tokens aus der vorgegebenen White-Space-Tokenisierung nie geändert werden.

Beispiel:

```
2x Amsterdam-Newcastle-Amsterdam inkl. Frühstückbuffet    € 244,80
2-Bett Innenkabine, Etagenbett
2x Bus Amsterdam CS-Terminal                               € 16,-
2x Bus CS-Terminal Amsterdam                               € 16,-
Kosten     DFDS                                           € 276,80
2x Bahntickets Köln – Amsterdam – Köln                     € 116,-
Deutsche Bahn Europa Spezial
pro Person                                                 € 196,40
```

Wenn die tabellenartige Struktur erhalten werden soll, muss sie in einem Vorverarbeitungsschritt durch entsprechende XML- bzw. HTML-Tags markiert werden. Diese werden gem. Abs. 4.6 unverändert als separate Tokens segmentiert. Sollten Posting- oder Absatzgrenzen sowie klar identifizierbare vom Text abgesetzte Dokumentbestandteile wie Bildunterschriften, Tabellen(zeilen), Aufzählungen etc. nicht durch Leerzeilen abgetrennt sein, so sind entsprechende Leerzeilen bei der manuellen Tokenisierung zu ergänzen. In der oben stehenden Tabelle würde dementsprechend jede Zeile durch eine Leerzeile abgetrennt.

5. Einfügen von Kommentaren

Bei der manuellen Nachbearbeitung der Tokenisierung kann es bisweilen hilfreich sein, zu problematischen Fällen einen Kommentar direkt in die Datei bzw. hinter ein Token zu schreiben. In bestimmten Situationen werden solche Kommentare von der vorliegenden Tokenisierungsrichtlinie explizit verlangt.

Um einen Kommentar zu einem Token anzufügen, geben Sie bitte direkt nach dem betreffenden Token ein Tabulatorzeichen ein (keine Serie von Leerzeichen!), danach zwei Prozentzeichen (`%%`) und anschließend Ihren Kommentar als Freitext.

Beispiele:

```
schonm  %% typo
al      %% typo
:--     %% könnte auch ein Smilie mit besonders langer Nase
        sein
```

Verwenden Sie innerhalb Ihres Kommentars bitte auf keinen Fall einen Zeilenwechsel (d.h.: betätigen Sie nirgendwo innerhalb des Kommentars die ENTER-Taste). Es ist kein Problem,

wenn Ihr Textverarbeitungsprogramm lange Zeilen automatisch umbricht und über mehrere Zeilen verteilt darstellt; Sie sollten aber sicherstellen, dass dabei keine Zeilenwechsel eingefügt werden.