Lösungsvorschlag für das Übungsblatt 6.

Aufgabe 1

b)

I.Informative Prosa:

A. Reportagen (Politik, Sport, Gesellschaft, Finanzen, Kultur)

B. Editorial (Personen-, Firmenbezogene Texte, Lernhilfe zu Programmen)

C. Reviews (Theater, Bücher, Musik, Tanzen)

D. Religion (Bücher, Zeitschriften, Traktate)

E. Fähigkeiten und Hobbies (Bücher, Zeitschriften)

F. Popular Lore (Bücher, Zeitschriften)

G. Belles Lettres, Biography, Memoirs etc. (Bücher, Zeitschriften)

H. Miscellaneous (Regierungsdokumente, Industrieberichte etc.)

J. Learned (Medizin, Mathematik, Politik etc.)

II. Imaginative Prosa:

K. General Fiction (Novellen/Romane oder Kurzgeschichten)

L. Mystery and Detective Fiction

M. Science Fiction

O. Adventure and Western Fiction

P. Romance und Lovestory

R. Humor

c)

jj – Adjektiv (Positiv)

cs – unterordnende Konjunktion

bedz – 3.Pers. Sg. von "be" in der Vergangenheitsform

np – Eigenname im Sg.

np$ - Eigenname mit dem possessiven Suffix "-'s" (possessiver Eigenname)

nn – Nomen im Sg.

pp$ - possessives Personalpronomen

bez – 3. Pers. Sg. von "be": "is"

ql – Partikel, die Adjektive verstärken

to – "to" in Infinitiv-Konstruktionen

vb – Verb

at – Artikel

wdt – "wh"-Determinator

pps – Personalpronomen, das mit der "-s"-Verbform im Präsens kongruiert

rb – Adverb

hvz – 3.Pers. Sg. von "have": "has"

cc – koordinierende Konjunktion

hv –"have"

vbn – Verb im Partizip Perfekt

in – Präposition

vbz – Verb in 3.Pers. Sg. Präsens

ap – Postdeterminator

ppo – Personalpronomen als Objekt (in einem Satz)

nns – Nomen im Plural

ber – für die Form "are" von "be"

md – Modalverbs

*- "not"

ben – Past Perfekt von "be": "been"

pp$$ - nominales Posessivpronomen (second nominal possessives)

nr – Adverbialnomen

dts – Demonstrativpronomen im Plural

vbd – Verb in der Vergangenheitsform

cd – Kardinalzahl

wrb – "wh"-Adverb

nr-tl – Adverbialnomen - Titelkonstituente

Interpunktionszeichen werden genauso geschrieben.

d) Kontextmuster zum Auflösen der Ambiguität:

1) nach np: Verb in der 3.Pers.

2) nach at: Nomen oder Gruppe3

3) nach pp$ oder np$: Nomen oder Gruppe4

4) nach jj: jj* (* für beliebig oft) Nomen

5) nach in: Nomen oder Gruppe2

6) nach to oder md: Verb in der Grundform

7) nach ql: Adjektiv

e)

1.Zusammengesetzte Wörter, deren Elemente durch space getrennt sind: z.B. plasma generator, have found, was covered, read into.

2. Zusammengesetzte Wörter, zwischen Elementen deren andere Einheiten stehen.

3. Stemming: z.B. was, taken, found.

4. Hyphenation: z.B. high-speed

5. Eigennamen: z.B. Pyrometer Instrument Co. Model 95

Aufgabe 2.

a)

Das Programm erkennt die Komposita, die jeweils aus zwei einfachen Nomen ohne Fugenelement bestehen. Das Lexikon enthält Nomen im Sg- und Pl-Form.

```
findSplitting(s) {
for i = 1..i = n do
    T1 = s0..si-1
    #prüft ob T1 ein LE (Lexikoneintrag) ist
    if L(T1) == LE then
        T2 = si..sn
        #prüft ob T2 ein LE ist
        if L(T2) == LE then
            push splittings, T1_ T2
        end if
    end if
end for
return splittings

}
```

b) Anzahl der Vergleiche hängt davon ab, an welcher Stelle das zu vergleichende Wort im Lexikon steht. Maximal Vergleiche: explizit für das Wort "Blumentopf" – 1.200.000 Vergleiche .

5 x 100.000 Vergleiche für "Blume" (1. if-Schleife)

100.000 – für "ntopf" (2.if-Schleife)

100.000 – für "Blumen" (1.if-Schleife)

100.000 – für "Topf" (2.if-Schleife)

4*100.000 – für den restlichen String "topf" (1.if-Schleife)

Einsparmöglichkeiten (Achtung, nur Anregungen zum Nachdenken): Lexikon nach Alphabet sortieren – String mit dem Lexikonstring an Stelle 50001 vergleichen, ist der Suchstring lexikalisch größer -> in der zweiten Hälfte suchen, sonst in der ersten. Usw. bis String gefunden ist. Rechnen Sie die Anzahl der Vergleiche jetzt für ein Lexikon mit 100 000 Eingriffen aus.

Lexikon in einen Hash speichern: was ist das? Jeder String wird mit einer Hashfunktion auf einen Zahlencode abgebildet und in eine Datenstruktur gespeichert, die Zugriff über diesen Zahlencode erlaubt.

Lexikon in einen sogenannten Trie speichern: Zugriff über die Buchstaben des Eingabestrings. Verzweigung pro Zustand mit Übergängen für das gesamte Alphabet.

Aufgabe 3.

1. ([A-Za-z]\.): z.B. "p.", "W.", "F.", "A.", "B."
2. ([A-Za-z]\.([A-Za-z0-9]\.)+): z.B. N.Y., U.S., S.A.,R.N.
3. ([A-Z][bcdfghj-np-tvxz]+\.): z..B. Mrs., Mr., Pp., Sgt., Is., Inc., Pp., St.
4. ([A-Za-z][^]*\.([,?;] [a-z0-9])): z.B. "Soc.Sec., p", "D.C., 1", "al., e", "al., e"
5. ([A-Za-z]([A-Za-z0-9])+): z.B. "words", "R09", "December", "Pp"
6. ([A-Z]): z.B. "A", "B", "J", "M"

Die Zeile im Dokument wieder finden:

z.B. für die erste Regel: s/($1)/<ABK1 $1>/g;

- die Abbreviatur wird in eckige Klammern "<ABK1 [Abbreviatur] >" gesetzt, so dass sie z.B. in jedem Editor durch Eingeben im Suchfeld "ABK1" wieder gefunden werden kann.

- "$1" steht für den Match, der in Klammern steht ([A-Za-z]\.).