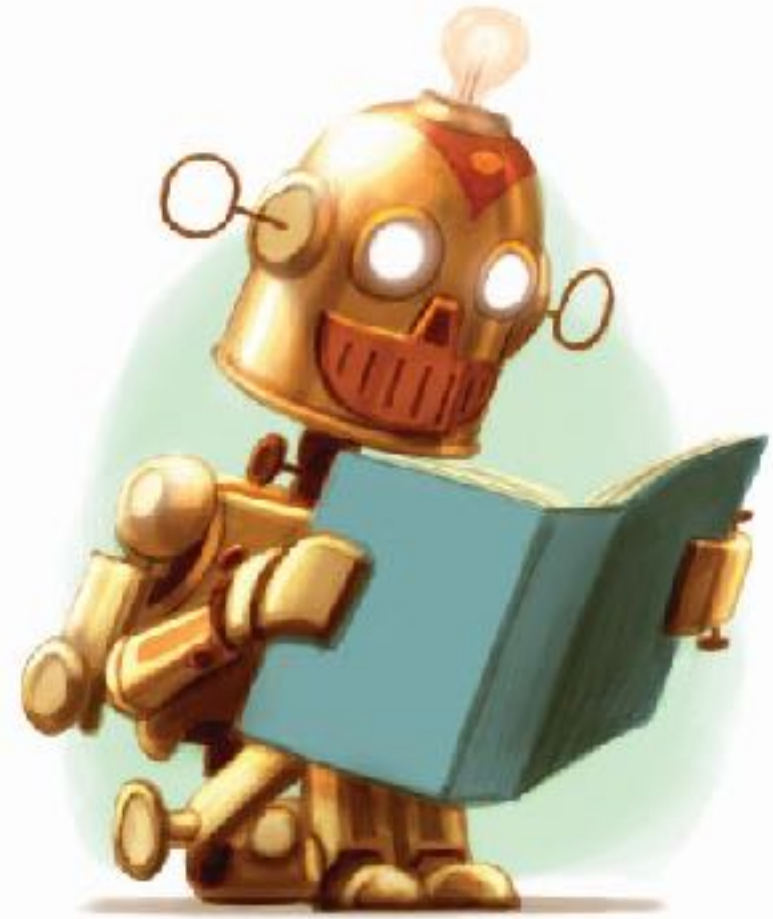
Text Mining

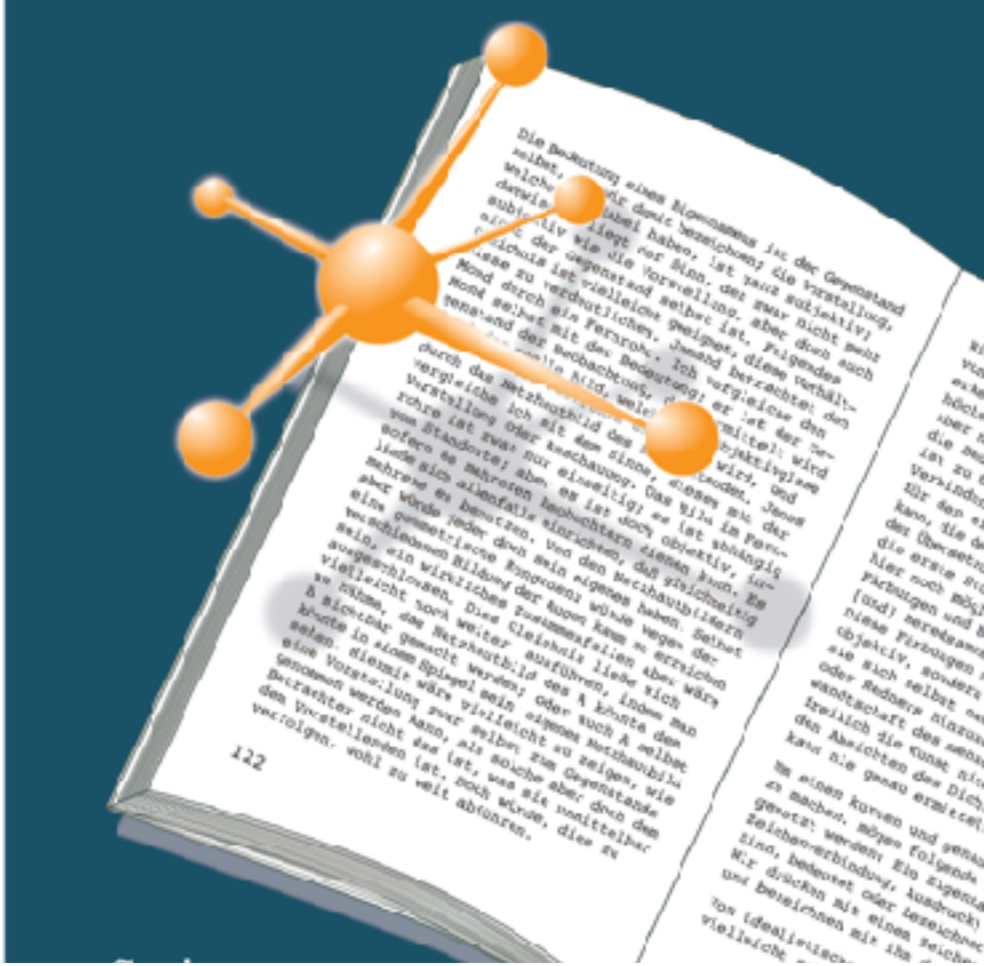


Natural Language Processing &
Semantic Computing

Prof. Dr. Siegfried Handschuh

UNIVERSITÄT PASSAU

Text Mining



Basic Text Processing

Prof. Dr. Siegfried Handschuh

Basic Text Processing

- Regular Expression,
- Regular Expressions in Practical NLP
- Word Tokenisation
- Word Normalisation and Stemming,
- Sentence Segmentation

Exercises

- Due to huge demand, we will have one big Exercises session!
- Bernhard will explain
- Individual Feedback via Piazza Learning System (you will get an introduction to that)

Regular Expressions

Regular expressions

- A formal language for specifying text strings
- How can we search for any of these?
  - woodchuck
  - woodchucks
  - Woodchuck
  - Woodchucks

Regular Expressions: Disjunctions

Letters inside square brackets []

Pattern	Matches
[wW]oodchuck	*Woodchuck, woodchuck*
[1234567890]	*Any digit*

Ranges [A-Z]

Pattern	Matches	
[A-Z]	*An upper case letter*	*Drenched Blossoms*
[a-z]	*A lower case letter*	*my beans were impatient*
[0-9]	*A single digit*	*Chapter 1: Down the Rabbit Hole*

Regular Expressions: Negation in Disjunction

Negations `[^Ss]`

Caret means negation only when first in []

Pattern	Matches	
`[^A-Z]`	*Not an upper case letter*	`Oyfn pripetchik`
`[^Ss]`	*Neither 'S' nor 's'*	`I have no exquisite reason"`
`[^e^]`	*Neither e nor ^*	`Look here`
`a\^b`	*The pattern a caret b*	`Look up a^b now`

Regular Expressions: More Disjunction

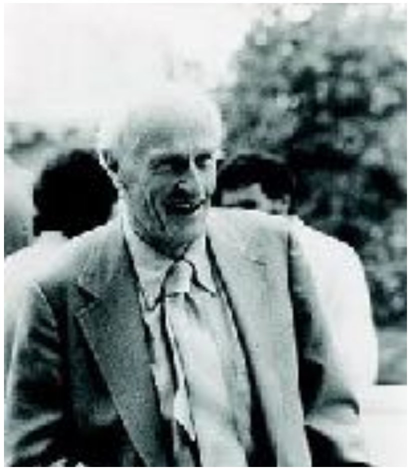
- Woodchuck is another name for groundhog!
- The pipe | for disjunction

Pattern	Matches
groundhog|woodchuck	
yours|mine	yours mine
a|b|c	= [abc]
[gG]roundhog| [Ww]oodchuck	

Regular Expressions:

? * + .

Pattern	Matches	
colou?r	*Optional previous char*	color colour
oo*h!	*0 or more of previous char*	oh! ooh! oooh! ooooh!
o+h!	*1 or more of previous char*	oh! ooh! oooh! ooooh!
baa+		baa baaa baaaa baaaaa
beg.n		begin begun begun beg3n



Stephen C Kleene

Kleene *, Kleene +

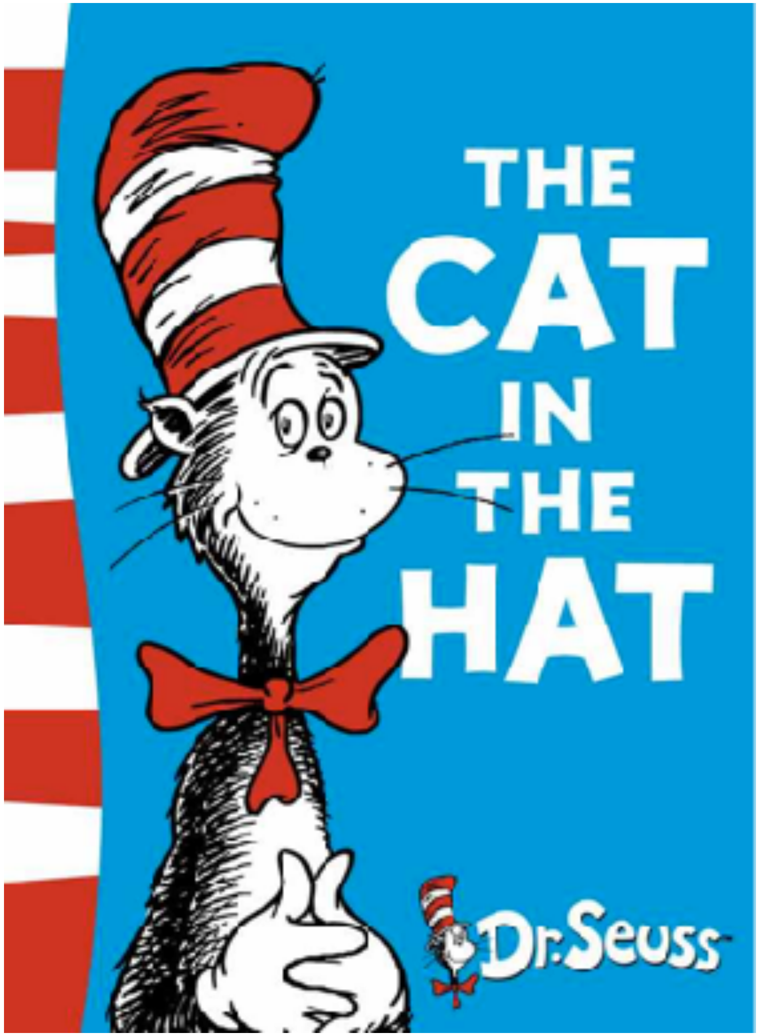
Regular Expressions: Anchors

^ $

Pattern	Matches
`^[A-Z]`	Palo Alto
`^[^A-Za-z]`	1 "Hello"
`\.$`	The end.
`.$`	The end? The end!

Excursus: Dr Seuss

Seuss's cat in the hat



Example: Cat in the Hat

We looked!
Then we saw him step in on the mat.
We looked!
And we saw him!
The Cat in the Hat!

The other one there, the blithe one.

Find me all instances of the word “the” in a text.

`the` Misses capitalised example

`[tT]he` Incorrectly returns other or blithe

`[^a-zA-Z][tT]he[^a-zA-Z]`

Errors

The process we just went through was based on fixing two kinds of errors

- Matching strings that we should not have matched (there, then, other)
    - False positives (Type I)
- Not matching things that we should have matched (The)
    - False negatives (Type II)

Errors cont.

- In NLP we are always dealing with these kinds of errors.
- Reducing the error rate for an application often involves two antagonistic efforts:
  - Increasing accuracy or precision (minimising false positives)
  - Increasing coverage or recall (minimising false negatives).

Summary

- Regular expressions play a surprisingly large role
  - Sophisticated sequences of regular expressions are often the first model for any text processing text
- For many hard tasks, we use machine learning classifiers
  - But regular expressions are used as features in the classifiers
  - Can be very useful in capturing generalisations

Assignment

- write regular expressions that extract **phone numbers** and regular expressions that extract **email addresses**.
- i.e. peter.mueller@uni-passau.de
- + also variations, such as:
  peter dot mueller at uni-passau dot de, etc.
- + java script encodings, such as used by the university of passau

Word tokenisation

Text Normalisation

- Every NLP task needs to do text normalisation:
  1. Segmenting/tokenising words in running text
  2. Normalising word formats
  3. Segmenting sentences in running text

How many words?

- I do uh main- mainly business data processing
  - Fragments, filled pauses
- Seuss's cat in the hat is different from other cats!
  - **Lemma**: same stem, part of speech, rough word sense
    - cat and cats = same lemma
  - **Wordform**: the full inflected surface form
    - cat and cats = different wordforms

How many words?

they lay back on the San Francisco grass and looked at the stars and their

- **Type**: an element of the vocabulary.
- **Token**: an instance of that type in running text.
- How many?
  - 15 tokens (or 14)
  - 13 types (or 12) (or 11?)

QUIZ

1. How many wordform types and tokens are in this sentence:

   I want to cook a dish that I never cooked before

   (A) 10 types 11 tokens

   (B) 11 types 10 tokens

   (C) 11 types 11 tokens

   (D) 11 types 10 tokens

How many words?

N = number of tokens

V = vocabulary = set of types

$|V|$ is the size of the vocabulary

	Tokens = N	Types = |V|
Switchboard phone	***2.4 million***	***20,000***
Shakespeare	***884,000***	***31,000***
Google N-grams	***1 trillion***	***13 million***

Simple Tokenization in UNIX

(Inspired by Ken Church's UNIX for Poets.)

Given a text file, output the word tokens and their frequencies

```
tr -sc 'A-Za-z' '\n' < shakes.txt
     | sort
     | uniq –c
```

Change all non-alpha to newlines

Sort in alphabetical order

Merge and count each type

```
   1
  17 A
   1 ADVENTURER
   1 ALL
   1 AND
   1 Above
   1 Accuse
   1 Admit
   1 Adonis
```

The first step: tokenising

```
tr -sc 'A-Za-z' '\n' < shakes.txt | head
```

```
THE
SONNETS
by
William
Shakespeare
From
fairest
creatures
We
...
```

The second step: sorting

```
tr -sc 'A-Za-z' '\n' < shakes.txt | sort | head

A
A
A
A
A
A
A
A
A
...
```

More Counting

Merging upper and lower case

```
tr 'A-Z' 'a-z' < shakes.txt | tr -sc 'A-Za-z' '\n' | sort
```

Sorting the counts

```
tr 'A-Z' 'a-z' < shakes.txt | tr -sc 'A-Za-z' '\n' | sort |
uniq -c | sort  -n -r
```

```
 322 that
  266 thy
  235 thou
  200 s
  195 love
  181 with
  172 for
  170 is
  168 not
  166 d
```

What happened here?

QUIZ

2. We saw that the words 'd' and 's' appear frequently in the word tokenization results. Which of following reasons might explain this?

   (A) The words "s" and "d" are archaic words that were more popular during the time that Shakespeare was writing.

   (B) The words "s" and "d" were not very common during the time that Shakespeare was writing, but they are among the many words which Shakespeare appears to have made up himself.

   (C) The tokenizer splits on all non-alphabetical characters, which includes the apostrophe. This mean that words like "Ophelia's" and '"dimm'd" will be tokenized to "Ophelia s" and "dimm d".

   (D) The Tokenizer splits words into smaller subparts named morphemes. "s" and "d" are two very common such morphemes because they tend to occur at the end of words.

Shakespeare

Will be a tatter'd weed,

Then being ask'd where

For where is she so fair whose unear'd womb

But if thou live, remember'd not to be,

Issues in Tokenisation

- `Finland's capital` → `Finland Finlands Finland's ?`
- `what're, I'm, isn't` → `What are, I am, is not`
- `Hewlett-Packard` → `Hewlett Packard ?`
- `state-of-the-art` → `state of the art ?`
- `Lowercase` → `lower-case lowercase lower case ?`
- `San Francisco` → one token or two?
- m.p.h., PhD. → ??

Tokenisation: language issues

- French
  - **L'ensemble** → one token or two?
  - **L** ? **L’** ? **Le** ?
  - Want **l’ensemble** to match with **un ensemble**
- German noun compounds are not segmented
  - **Versicherungsangestellter** =
    insurance company employee
  - **Donaudampfschifffahrtskapitän** =
    danube steamship captain’
  - German information retrieval needs compound splitter

Tokenisation: language issues

- Chinese and Japanese no spaces between words:
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
  - Sharapova now lives in US southeastern Florida
- Further complicated in Japanese, with multiple alphabets intermingled
  - Dates/amounts in multiple formats

フォーチュン500社は情報不足のため時間あた$500K(約6,000万円)

Katakana Hiragana Kanji Romaji

End-user can express query entirely in hiragana!

Word Tokenisation in Chinese

- Also called **Word Segmentation**
- Chinese words are composed of characters
  - Characters are generally 1 syllable and 1 morpheme.
  - Average word is 2.4 characters long.
- Standard baseline segmentation algorithm:
  - Maximum Matching (also called Greedy)

Maximum Matching Word Segmentation Algorithm

- Given a wordlist of Chinese, and a string.

1. Start a pointer at the beginning of the string
2. Find the longest word in dictionary that matches the string starting at pointer
3. Move the pointer over the word in string
4. Go to 2

Max-match segmentation illustration

- Thecatinthehat — the cat in the hat
- Thetabledownthere — the table down there

  theta bled own there

- Doesn't generally work in English!

- But works astonishingly well in Chinese
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
- Modern probabilistic segmentation algorithms even better

Word Normalisation and Stemming

Normalisation

- Need to "normalise" terms
  - Information Retrieval: indexed text & query terms must have same form.
    - We want to match ***U.S.A.*** and ***USA***
- We implicitly define equivalence classes of terms
  - e.g., deleting periods in a term
- Alternative: asymmetric expansion:
  - Enter: ***window*** Search: ***window, windows***
  - Enter: ***windows*** Search: ***Windows, windows, window***
  - Enter: ***Windows*** Search: ***Windows***
- Potentially more powerful, but less efficient

Case folding

- Applications like IR: reduce all letters to lower case
  - Since users tend to use lower case
  - Possible exception: upper case in mid-sentence?
    - e.g., ***General Motors***
    - ***Fed*** vs. ***fed***
    - ***SAIL*** vs. ***sail***
- For sentiment analysis, MT, Information extraction
  - Case is helpful (***US*** versus ***us*** is important)

Lemmatisation

- Reduce inflections or variant forms to base form
  - *am, are, is* → *be*
  - *car, cars, car's, cars'* → *car*
- *the boy's cars are different colours* → *the boy car be different colour*
- Lemmatisation: have to find correct dictionary headword form
- Machine translation
  - Spanish quiero ('I want'), quieres ('you want') same lemma as querer 'want'

Morphology

- **Morphemes**:
  - The small meaningful units that make up words
  - **Stems**: The core meaning-bearing units
  - **Affixes**: Bits and pieces that adhere to stems
    - Often with grammatical functions

Stemming

- Reduce terms to their stems in information retrieval
- *Stemming* is crude chopping of affixes
  - language dependent
  - e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.

for example compressed and compression are both accepted as equivalent to compress.

→

for exampl compress and compress ar both accept as equival to compress

Porter's algorithm: The most common English stemmer

Step 1a

```
sses → ss      caresses → caress
ies  → i       ponies   → poni
ss   → ss      caress   → caress
s    → ø       cats     → cat
```

Step 1b

```
(*v*)ing → ø   walking   → walk
               sing      → sing
(*v*)ed  → ø   plastered → plaster
…
```

Step 2 (for long stems)

```
ational→ ate   relational→ relate
izer→ ize      digitizer → digitize
ator→ ate      operator  → operate
…
```

Step 3 (for longer stems)

```
al    → ø   revival    → reviv
able  → ø   adjustable → adjust
ate   → ø   activate   → activ
…
```

QUIZ

3. Given the description you saw on earlier slides, the Porter stemmer would stem the word 'aching' as

   (A) aching

   (B) ach

   (C) ache

   (D) aches

Why only strip –ing if there is a vowel?

```
(*v*)ing → ø   walking   → walk
               sing      → sing
```

Viewing morphology in a corpus
Why only strip –ing if there is a vowel?

(*V*)ing → ø walking → walk

sing → sing

```
tr -sc 'A-Za-z' '\n' < shakes.txt | grep 'ing$' | sort | uniq -c | sort –nr
```

```
tr -sc 'A-Za-z' '\n' < shakes.txt | grep '[aeiou].*ing$' | sort | uniq -c | sort –nr
```

1312 King	548 being
548 being	541 nothing
541 nothing	152 something
388 king	145 coming
375 bring	130 morning
358 thing	122 having
307 ring	120 living
152 something	117 loving
145 coming	116 Being
130 morning	102 going

Viewing morphology in a corpus
Why only strip –ing if there is a vowel?

(*v*)ing → ø walking → walk
sing → sing

```
tr -sc 'A-Za-z' '\n' < shakes.txt | grep 'ing$' | sort | uniq -c | sort -nr
```

1312	King	548	being
548	being	541	nothing
541	nothing	152	something
388	king	145	coming
375	bring	130	morning
358	thing	122	having
307	ring	120	living
152	something	117	loving
145	coming	116	Being
130	morning	102	going

```
!tr -sc 'A-Za-z' '\n' < shakes.txt | grep '[aeiou].*ing$' | sort | uniq
-c | sort -nr
```

Dealing with complex morphology is sometimes necessary

- Some languages requires complex morpheme segmentation
  - Turkish
  - Uygarlastiramadiklarimizdanmissinizcasina
  - `(behaving) as if you are among those whom we could not civilised'
  - Uygar `civilized' + las `become'

    + tir `cause' + ama `not able'

    + dik `past' + lar 'plural'

    + imiz 'p1pl' + dan 'abl'

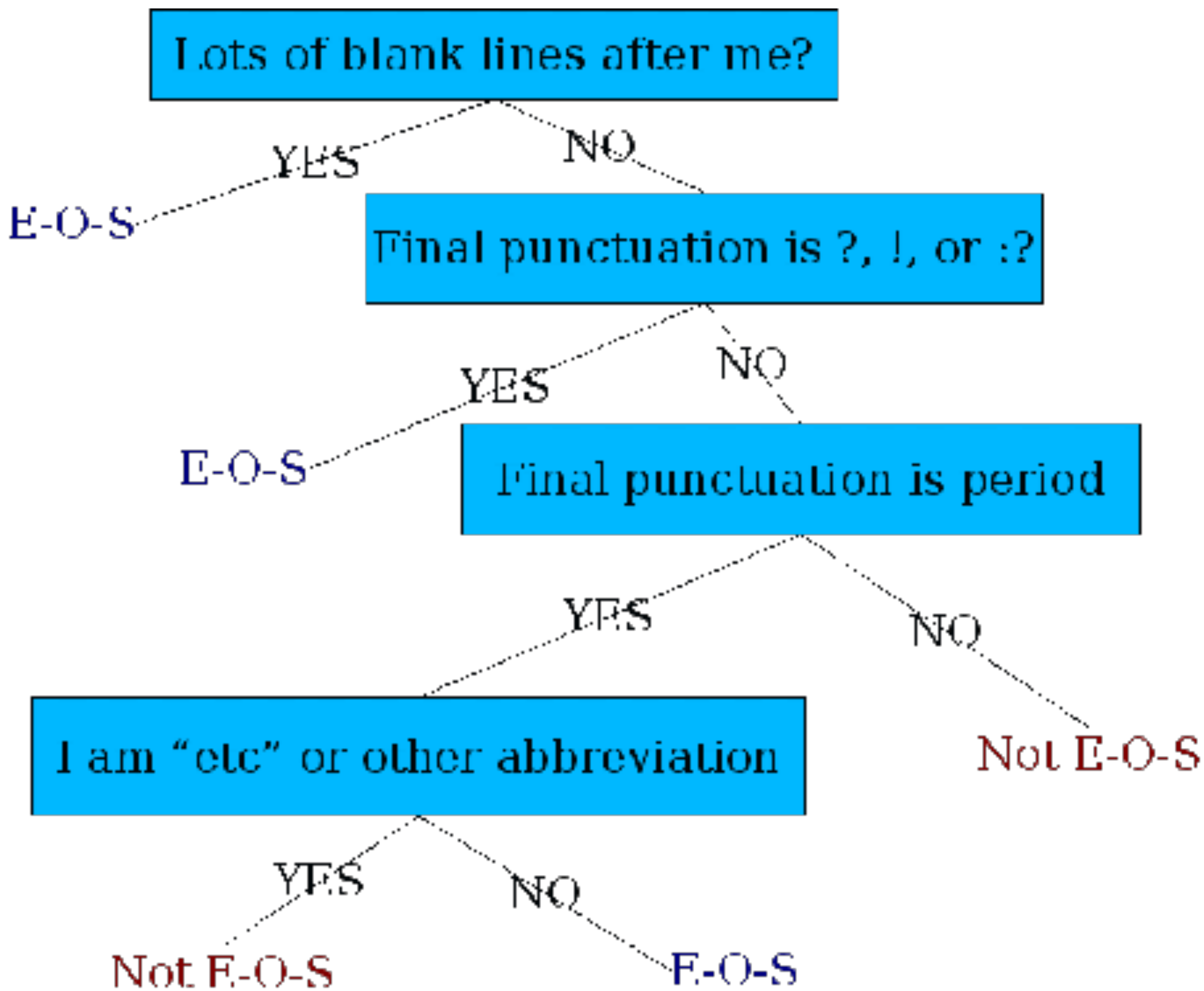
    + mis 'past' + siniz '2pl' + casina 'as if'

Sentence Segmentation

Sentence Segmentation

- !, ? are relatively unambiguous
- Period "." is quite ambiguous
  - Sentence boundary
  - Abbreviations like Inc. or Dr.
  - Numbers like .02% or 4.3
- Build a binary classifier
  - Looks at a "."
  - Decides EndOfSentence/NotEndOfSentence
  - Classifiers: hand-written rules, regular expressions, or machine-learning

Determining if a word is end-of-sentence: a Decision Tree



More sophisticated decision tree features

- Case of word with ".": Upper, Lower, Cap, Number
- Case of word after ".": Upper, Lower, Cap, Number

- Numeric features
  - Length of word with "."
  - Probability(word with "." occurs at end-of-s)
  - Probability(word after "." occurs at beginning-of-s)

QUIZ

A period (".") occurs at the end of four words that each have one of the following four wordshapes. Which of the four periods (all else being equal) is more likely than the other to be a sentence boundary?

(A) Upper

(B) Lower

(C) Cap

(D) Number

QUIZ

- A: “… Word.”
- B: “… word.” most likely to be EOS (End Of Sentence)
- C “….WORD.”
- D” … 9.”

Implementing Decision Trees

- A decision tree is just an if-then-else statement
- The interesting research is choosing the features
- Setting up the structure is often too hard to do by hand
  - Hand-building only possible for very simple features, domains
    - For numeric features, it's too hard to pick each threshold
  - Instead, structure usually learned by machine learning from a training corpus

Decision Trees and other classifiers

- We can think of the questions in a decision tree
- As features that could be exploited by any kind of classifier
  - Logistic regression
  - SVM
  - Neural Nets
  - etc.

Basic Text Processing

- Regular Expression,
- Regular Expressions in Practical NLP
- Word Tokenisation
- Word Normalisation and Stemming,
- Sentence Segmentation