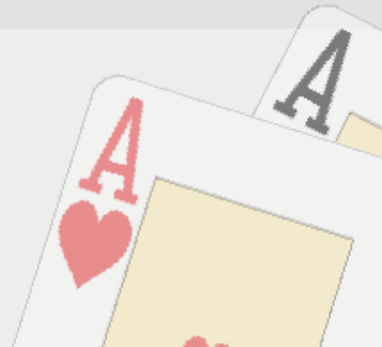
AI Games course

Certificate 2, session 2
Classification of textual data.
Linguistic features.



Language Identification

- **Task**: given a phrase in an unknown Germanic language, identify the language

- Languages: Dutch, Norwegian, Swedish

- Example:

“gelukkige verjaardag”	→	Dutch
“Gratulerer med dagen”	→	Norwegian
“Grattis på födelsedagen”	→	Swedish

Step 1: Data instances

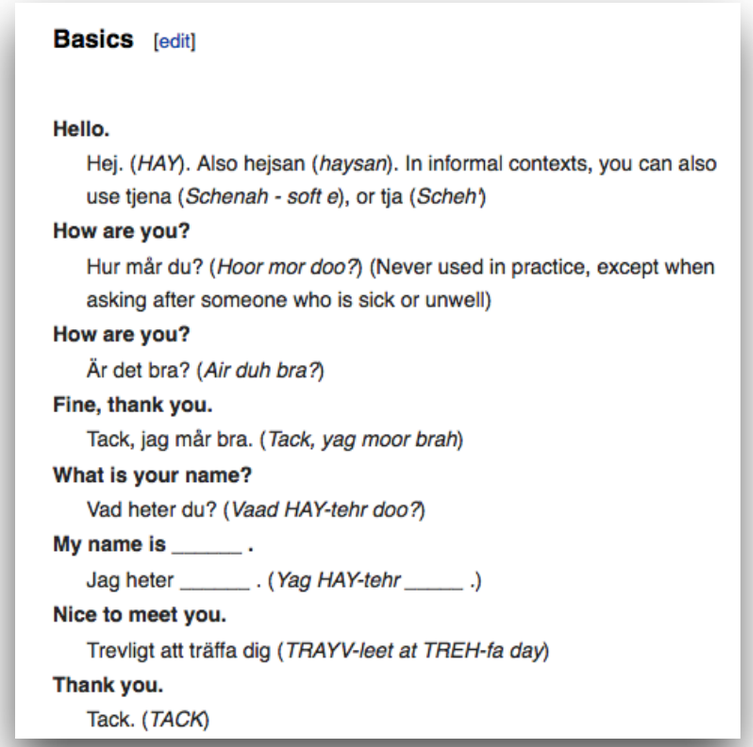
- we use Wikitravel phrasebooks: https://wikitravel.org/en/List_of_phrasebooks

- download *phrases.txt*

- file structure:

  % comment line with translation into English

  phrase in a foreign language ||| LANGUAGE_CODE



Basics [edit]

Hello.
Hej. (*HAY*). Also hejsan (*haysan*). In informal contexts, you can also use tjena (*Schenah - soft e*), or tja (*Scheh'*)
How are you?
Hur mår du? (*Hoor mor doo?*) (Never used in practice, except when asking after someone who is sick or unwell)
How are you?
Är det bra? (*Air duh bra?*)
Fine, thank you.
Tack, jag mår bra. (*Tack, yag moor brah*)
What is your name?
Vad heter du? (*Vaad HAY-tehr doo?*)
My name is _____ .
Jag heter _____ . (*Yag HAY-tehr _____ .*)
Nice to meet you.
Trevligt att träffa dig (*TRAYV-leet at TREH-fa day*)
Thank you.
Tack. (*TACK*)

Languages: *SWE (Swedish), NOR (Norwegian), DUT (Dutch)*

Step 2: Linguistic features

- **language model** — probability distribution over sequences of words or other linguistic units

- **ngram** — a continuous sequence of *n* items from a text sample

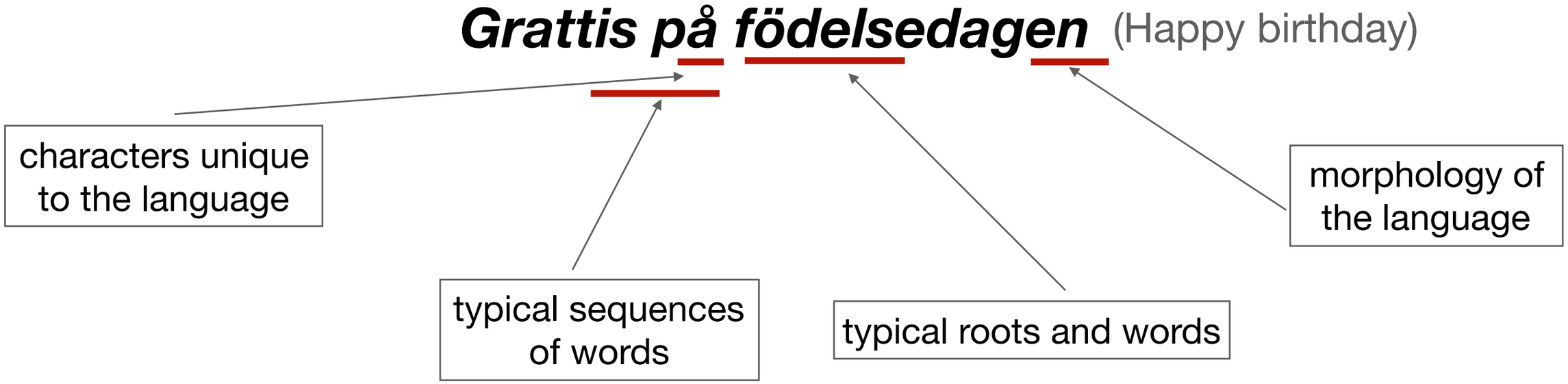
- **Example**: '*how to make an AI gaming bot*'
  - word trigrams:

    '*how to make*', '*to make an*', '*make an AI*', '*an AI gaming*', '*AI gaming bot*'
  - character trigrams

    '*how*', '*ow '*, '*w t*', '*' to*', '*to '*, '*o m*', '*' ma*', '*mak*', '*ake*', '*ke '*, '*e a*', '*' an*', '*an '*, '*'n A*', '*' AI*', '*AI '*, '*I g*', '*' ga*', '*gam*', '*ami*', '*min*', '*ing*', '*ng '* etc.

Why ngrams work

Ngrams (up to length 5) are a surprisingly powerful and universal tool to model language. They can capture:



Extracting features

```
# feature extraction

s = "Kunt u mij dat tonen op de kaart"

ngrams = []
n = 3 # size of ngrams
for i in range(len(s) - n + 1):
    ngram = s[i:i+n]
    ngrams.append(ngram)

print(ngrams)
```

Task 1: modify the code so that it generates ngrams of length *up to n*
Task 2: write code for extracting word ngrams

Extracting features

Things to think about:

- combine word and character ngrams
- vary ngram size (parameter *n*)
- upper-case and lower-case letters [*s.lower()*]
- extra spaces [*s.strip()*]
- punctuation signs [*s.replace(“,”, “”)*]

Step 3: Learning algorithm

- there are hundreds of different machine learning algorithms
    - for classification/regression/clustering
    - for supervised/unsupervised/reinforcement learning
    - for various types of data
- today, we are going to use *logistic regression*