Statistical Natural Language Processing:

N-GRAM MODELS

Sveta Zinger

s.zinger@rug.nl

Seminar in Methodology and Statistics

Rijksuniversiteit Groningen

15 March 2006

... the united states of ???

$P(\text{states} \mid \text{the united}) = \ldots$

$P(\text{of} \mid \text{united states}) = \ldots$

$P(\text{America} \mid \text{states of}) = \ldots$

$P(\text{Belgium} \mid \text{states of}) = \ldots$

Statistical inference

language modeling:

predict the next word given the previous words

Applications

- handwriting recognition
- speech recognition
- optical character recognition
- spelling correction
- machine translation

Predicting the next word is estimating the probability function P:

$$P(w_n|w_1,\dots,w_{n-1})$$

w is a word, *n* – its number in a sequence

Markov assumption:

only the prior local context – the last few words – affects the next word

Usually used n-grams

$w_1 w_2$	bigram
$w_1 w_2 w_3$	trigram
$w_1 w_2 w_3 w_4$	four-gram

Importance of large n-gram models

... the large green { pill, frog / tree, car, mountain }

Sue swallowed the large green { pill, frog / ~~tree, car, mountain~~ }

Larger n-grams → more parameters to estimate

Possible ways to reduce the vocabulary for n-gram models

stemming
(removing the inflectional
endings from words)

grouping words into
semantic classes
(by pre-existing thesaurus
or by induced clustering)

Advantages of n-gram model: simple, easy to calculate, work well to predict words (trigrams, for example).

n-gram models work best when trained on large amounts of data

Probability of having the word w_n after the sequence of words $w_1 \ldots w_{n-1}$

$$P(w_n | w_1 \ldots w_{n-1}) = \frac{P(w_1 \ldots w_n)}{P(w_1 \ldots w_{n-1})}$$

Maximum Likelihood Estimate (MLE):

$$P_{MLE}(w_1 \ldots w_n) = \frac{C(w_1 \ldots w_n)}{N}$$

$C(w_1 \ldots w_n)$ - frequency of n-gram $w_1 \ldots w_n$ in training text,

N – number of training instances

$$P(w_n | w_1 \ldots w_{n-1}) = \frac{C(w_1 \ldots w_n)}{C(w_1 \ldots w_{n-1})}$$

Example: predict the word after the words *comes across*

Trigrams:

trigram starting by *comes across* occurred *N*=10 times

$C(comes\ \ across)=10$

comes across as
comes across as
comes across as
comes across as
comes across as
comes across as
comes across as
comes across as

$C(comes\ \ across\ \ as)=8$

comes across more

$C(comes\ \ across\ \ more)=1$

comes across a

$C(comes\ \ across\ \ a)=1$

$$P(as|comes\ \ across)=\frac{C(comes\ \ across\ \ as)}{C(comes\ \ across)}=0.8$$

$$P(more) = \frac{C(comes\ \ across\ \ more)}{C(comes\ \ across)} = 0.1$$

$$P(a) = 0.1$$

If *x* is not among the three above words (*as*, *more*, *a*) then

$$P(x) = 0.0$$

MLE does not capture the fact that other words can follow *comes across,* like *the* and *some*

Discounting (smoothing) methods:

decrease the probability of previously seen events to leave some probability for previously unseen events

Better estimators

Laplace's law

$$P_{Lap}(w_1 \ldots w_n) = \frac{C(w_1 \ldots w_n) + 1}{N + B}$$

B – number of possible sequences. For unigrams *B* is *V* – vocabulary size,

for n-grams *B* is V^n

Laplace's law often gives too much of the probability space to unseen events

Lidstone's law:

$$P_{Lid}(w_1 \ldots w_n) = \frac{C(w_1 \ldots w_n) + \lambda}{N + B\lambda}$$

- λ has to be tuned
- probability estimates are linear in the MLE frequency

How much probability should be left for unseen events?

The held out estimator: $P_{ho}(w_1 \dots w_n) = \dfrac{T_r}{N_r N}$

where

$$T_r = \sum_{\{w_1 \dots w_n : C_1(w_1 \dots w_n) = r\}} C_2(w_1 \dots w_n)$$

$C_1(w_1 \dots w_n)$ - frequency of the n-gram in training data

$C_2(w_1 \dots w_n)$ - frequency of the n-gram in held out data

N_r - the number of n-grams with frequency r (in the training text)

T_r - the total number of times that all n-grams that appeared *r* times in the training text appeared in the held out data

Example; average frequency for the held out estimator

n-grams in the training text	frequency
a	5
b	3
c	2
d	2
e	2

n-grams in the held out text	frequency
f	10
g	7
h	5
d	3
e	3

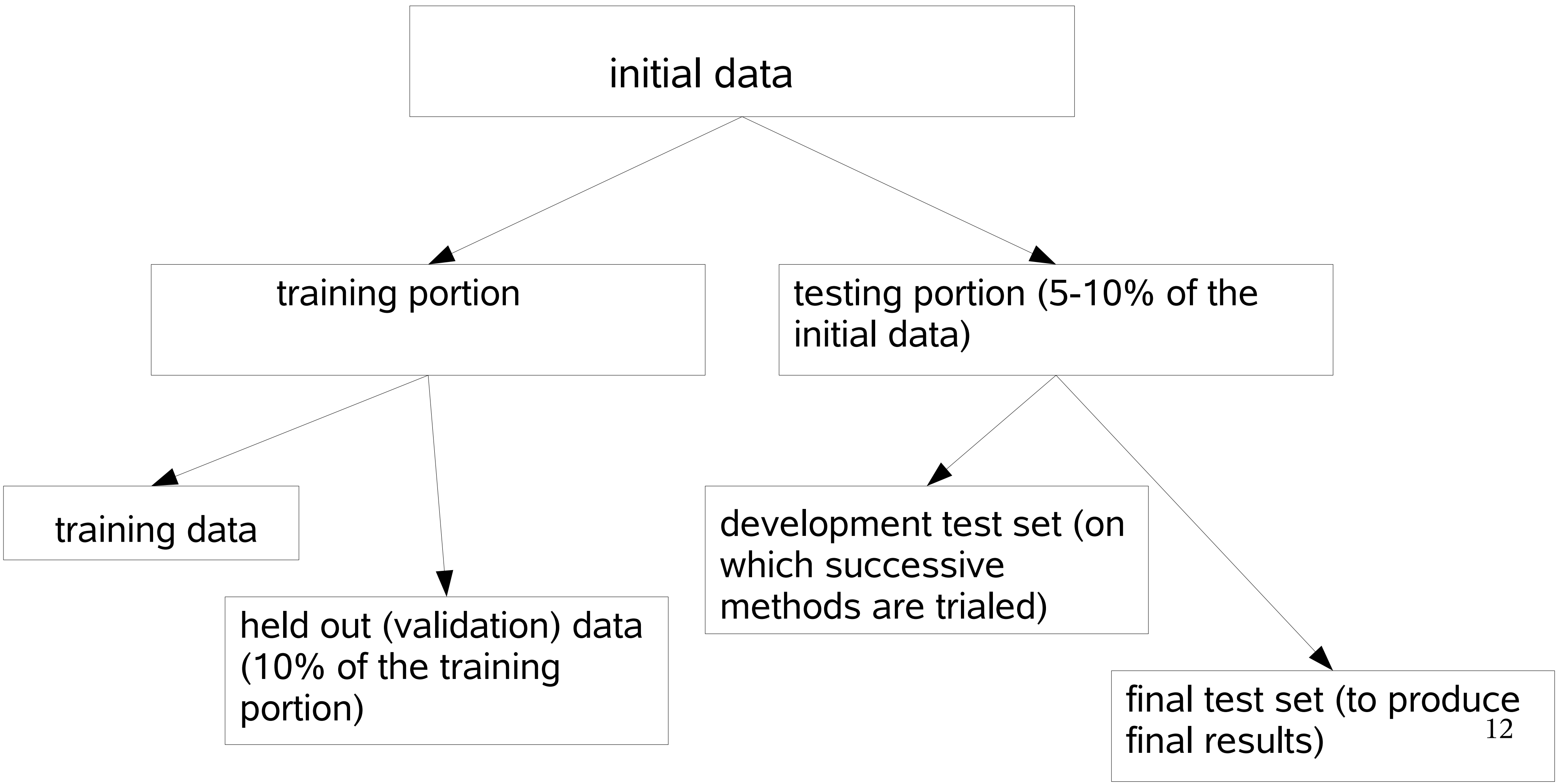
$r = 2 \quad N_r = 3$

$T_r = 6$

average frequency is $\dfrac{T_r}{N_r} = \dfrac{6}{2} = 3$

Data for training and testing models

models induced from a sample of data are often overtrained
--> test data should be independent from the training data



Which parts of the data are to be used as testing data?

- select bits (sentences or n-grams) randomly from throughout the data for the test set and use the rest of the material for training;

training set is a very good sample of the test data

- set aside large chunks as test data;

testing set is slightly different from the training set, better simulation of a real-life situation

Cross validation

each part of the training set is used both as initial training data and as held out data

Deleted estimation:

$$P_{ho}(w_1 \ldots w_n) = \frac{T_r^{01}}{N_r^0 N} \quad \text{or} \quad \frac{T_r^{10}}{N_r^1 N}$$

N_r^a - the number of n-grams occurring *r* times in the part of the training data

T_r^{ab} - the total occurrences of those n-grams from part *a* in the part *b*

Cross validation

Deleted interpolation

$$P_{del}(w_1 \ldots w_n) = \frac{T_r^{01} + T_r^{01}}{N\left(N_r^0 + N_r^1\right)}$$

Leaving-One-Out method

training corpus is of size *N-1* tokens, while one token is used as held out data for testing;

the process is repeated *N* times – each piece of data is left out in turn;

advantage – explores the effect of how the model changes if any piece of data had not been observed

Good-Turing estimator

determines adjusted frequency of items:

$$r^* = (r+1)\frac{E(N_{r+1})}{E(N_r)}$$

E – expectation of a random variable

How to get expectation ?

- use N_r instead if the expectation – works for low frequencies, then MLE can be applied for high frequencies
- fit some function *S* through the observed values (r, N_r)

  and use the values of *S(r)* for the expectation

Combining estimators

mix a trigram model with bigram and unigram models that suffer less from sparseness

Linear interpolation

$$P_{li}(w_n|w_{n-2},w_{n-1})=\lambda_1 P_1(w_n)+\lambda_2 P_2(w_n|w_{n-1})+\lambda_3 P_3(w_n|w_{n-1},w_{n-2})$$

Other combining estimators:

- Katz's backing off (recursive, uses progressively shorter histories)
- general linear interpolation (weights are a function of the history)

SCRATCH
(SCRipt Analysis Tools for the Cultural Heritage) project

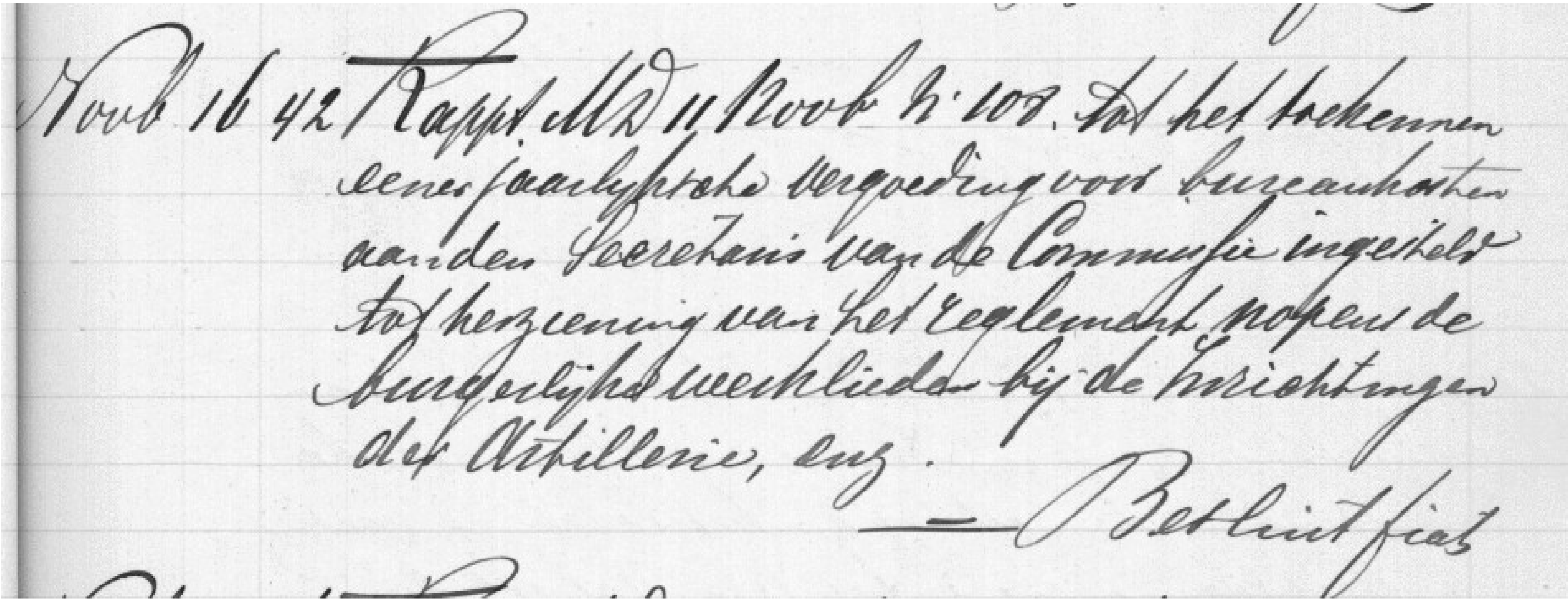
Data:

archive of Royal decrees (Kabinet der Koningin) – scanned pages of handwritten text, sometiems (rarely) annotated manually (ASCII text)

Goal:

enable search through the handwritten text like Google does for the texts in electronic form

Example of data



Novb 16 42 Rappt. MD 11 Novb no 108 tot het toekennen eener jaarlijksche vergoeding voor bureaukosten aan den Secretaris van de Commissie ingesteld tot herziening van het reglement nopens de burgerlijke werklieden bij de Inrichtingen der Artillerie, enz.

__

____Besluit fiat

Ideal case:

pattern recognition on handwritten text leads to imperfect phrases, later analysed and improved by a linguistic model (n-grams, for example)

Reality:

linguistic data have to be incorporated since the beginning to help recognize patterns in the handwritten text;

a model combining pixels and words has to be used (maybe similar to speech recognition)