Computational Linguistics 2014-2015

- **Walter Daelemans** (walter.daelemans@uantwerpen.be)
- **Guy De Pauw** (guy.depauw@uantwerpen.be)
- **Mike Kestemont** (mike.kestemont@uantwerpen.be)

http://www.clips.uantwerpen.be/cl1415

Universiteit Antwerpen

Practical

Location	P0.11 (Scribanihuis)
Reading material	• D. Jurafsky & J.H. Martin (2009) Speech and Language Processing - An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (2nd ed). Pearson Education, USA. • Natural Language Processing with Python
Software	Python 3.4 and NLTK: Installation Instructions
Evaluation	Take-home assignments and oral examination
Lecturers	Walter Daelemans: walter.daelemans@uantwerpen.be Mike Kestemont: mike.kestemont@uantwerpen.be Guy De Pauw: guy.depauw@uantwerpen.be

Program

Session	Day	Date	Chapter	Topic	Reading Assignment	Slides	Take-home Assignment
1	Monday	29/9/2014	**Python**	Session 1 - Variables	See Github		
2	Thursday	2/10/2014	**Python**	Session 2 - Collections			
3	Monday	6/10/2014	**Python**	Session 3 - Conditions (and an introduction to loops)			
4	Thursday	9/10/2014	**Python**	Session 4 - Loops			
5	Monday	13/10/2014	**Python**	Session 5 - Reading and writing to files			
6	Thursday	16/10/2014	**Python**	Session 6 - Writing your own Functions and importing packages			
7	Monday	20/10/2014	**Python**	Session 7 - Regular Expressions in Python			
8	Thursday	23/10/2014	**Python**	Session 8 - Advanced looping in Python and list comprehensions			
9	Monday	27/10/2014	**Theory**	Introduction to Computational Linguistics	Jurafsky & Martin: Chapter 1	PDF	
10	Monday	3/11/2014	**Theory**	Regular Expressions and Finite State Automata & Transducers	Jurafsky & Martin: Chapter 2; Chapter 3	Slides morfsegment.py	See last slide. Deadline: 24/11 participles.py
	Monday	10/11/2014	**Remembrance day: no session**				
11	Monday	17/11/2014	**Theory**	Part-of-Speech Tagging	Jurafsky & Martin: Chapter 5 (not 5.5, 5.8 and 5.9)	Slides Python Code	See last slide. Deadline: 8/12
12	Monday	24/11/2014	**Theory**	Syntactic Analysis & Parsing	Jurafsky & Martin: Chapter 12 (not 12.7.2, 12.8); Chapter 13 (not 13.4.1, 13.4.2, 13.5.1)		
13	Monday	1/12/2014	**Theory**	Minimum Edit Distance + Probabilistic Methods	Jurafsky & Martin: Chapter 3.11; Chapter 4.1, 4.2 and 4.3; Chapter 5.5 and 5.9; Chapter 14.1, 14.3 and 14.4;		
14	Monday	8/12/2014	**Theory**	Word Sense Disambiguation	Jurafsky & Martin: Chapter 19.1, 19.2, 19.3, Chapter 20 (20.1->20.5)		
15	Monday	15/12/2014	**Theory**	Sentence semantics and discourse; Information extraction	Jurafsky & Martin: Chapter 21; Chapter 22		

Bayesian Inference

N-gram models

Statistical Methods

- automatically derive statistical data from (annotated) corpora
- frequency of observed events are interpreted as the **probability** of those events occurring in the future
- We can use these probabilities to perform disambiguation

e.g. Most likely tag for *can* in “I can do this.”?

P(MD|can) *vs* P(NN|can) *vs* P(VB|can)

- P(x|y) is calculated through *Bayesian Inference*

Noisy Channel

INPUT → NOISY CHANNEL → OUTPUT

Task	Input	Output
Speech Recognition	String of Words	Acoustic Signal
OCR/ Spellchecking	Correct Text	Text with errors
POS Tagging	String of POS Tags	String of words
Machine Translation	Sentence in English	Sentence in Chinese

P(Input|Output) ?????

Bayesian Inference

Bayes' Law $P(x|y) = \frac{P(y|x).P(x)}{P(y)}$

e.g. From Wikipedia

Drug Test: 0.99 accurate (99% chance that a user tests positive, 99% chance that a non-user tests negative)

Users: 0.5% of the population

What is the probability that someone who tests positive, is a user?

$$P(User|+) = \frac{P(+|User).P(user)}{P(+)}$$

$$= \frac{0.99 * 0.005}{P(+|User)*P(User) + P(+|non\text{-}user)*P(non\text{-}user)}$$

$$= \frac{0.99 * 0.005}{0.99*0.005 + 0.01 * 0.995}$$

$$= 0.332$$

Bayesian Inference

Bayes' Law $P(x|y) = \frac{P(y|x).P(x)}{P(y)}$

From a corpus we calculated the following probabilities
P(can|MD) = 0.8 (the frequency with which *can* was observed as MD)
P(can|NN) = 0.1 and P(can|VB) = 0.1
P(MD) = 0.05, P(NN) = 0.3 and P(VB) = 0.1
P(can) = 0.00001

What is P(MD|can), the probability that we need to tag MD when we see 'can'?

$$P(MD|can) = \frac{P(can|MD).P(MD)}{P(can)}$$

$$= \frac{0.8 * 0.05}{1} = 0.04$$

P(NN|can) = P(can|NN).P(NN) = 0.1 * 0.3 = 0.03
P(VB|can) = P(can|VB).P(VB) = 0.1 * 0.1 = 0.01

Exercise

$$P(x|y) = \frac{P(y|x).P(x)}{P(y)}$$

"can" is counted 60 times as "MD" in corpusA and 40 times as "NN". In corpusB "can" is counted 70 times as "NN" and 30 times as "MD".

1. What is the probability of "can" as "MD" in corpusA?
2. What is the probability of "can" as "NN" in corpusB?
3. We pick a sentence randomly from one of the 2 corpora:

   "I can do this"

What is the probability that this sentence came from corpusA?

Exercise

$$P(x|y) = \frac{P(y|x).P(x)}{P(y)}$$

"can" is counted 60 times as "MD" in corpusA and 40 times as "NN". In corpusB "can" is counted 70 times as "NN" and 30 times as "MD".

1. What is the probability of "can" as "MD" in corpusA?
2. What is the probability of "can" as "NN" in corpusB?
3. We pick a sentence randomly from one of the 2 corpora:
   "I can do this"

What is the probability that this sentence came from corpusA?

P(corpusA|canMD) = (P(canMD|corpusA).P(corpusA)) / P(canMD)
= (60/100 x 1/2) / (60+30/200)
= (0.6x0.5) / 0.45
= 0.667

Bayesian Inference

In language technology, we calculate the probability of the association between an input sequence and an output sequence.

e.g. Machine translation

argmaxInput P(*fille*|*girl*) =

argmaxInput P(*fille*) P(girl|fille)

P(*fille*)	P(girl|fille)
Prior (Language Model)	Likelihood (Domain Model)

Bayesian Inference

argmaxInput P(*fille*|*girl*) =

argmaxInput P(*fille*) P(girl|fille)

Prior (Language Model)	Likelihood (Domain Model)

The Domain Model provides the probability that *girl* can be translated as *fille*
The language model provides the probability that the word *fille* exist (in that context)

Bayes’ Rule & Noisy Channel

	P(Input)	P(Output|Input)
Machine Translation	Language Model	Translation model
OCR		Model of OCR errors
Spellchecking		Model of spelling errors
POS-Tagging		Tag-Word Model
Speech Recognition		Acoustic model

Language Model

- What is the probability of a given sequence of words, tokens, tags?
- Most common: n-gram models
- data driven: given $n_1, n_2, n_3, n_4, \ldots n_z$
- unigram: $P(word) = freq(word) / N$

$P(sentence) = \prod P(word)$

Language Model

- What is the probability of a given sequence of words, tokens, tags?
- Most common: n-gram models
- data driven: given $n_1, n_2, n_3, n_4, \ldots n_z$
- unigram: $P(word) = freq(word) / N$

  $P(sentence) = \prod P(word)$

Language Model

- What is the probability of a given sequence of words, tokens, tags?
- Most common: n-gram models
- data driven: given $n_1, n_2, n_3, n_4, \ldots n_z$
- unigram: P(word) = freq(word) / N

  $P(\text{sentence}) = \prod P(\text{word})$

Language Model

- What is the probability of a given sequence of words, tokens, tags?
- Most common: n-gram models
- data driven: given $n_1, n_2, n_3, n_4, \ldots n_z$
- unigram: P(word) = freq(word) / N

  P(sentence) = ∏P(word)

Language Model

- But unigram is a weak language model
- Suppose we want to predict the most likely possible word in the sentence

Just then, the white…

According to unigram:

P(the) = 0.07 P(*rabbit)* = 0.00001

And so

P(Just then, the white the) > P(Just then, the white rabbit)

Although intuitively

P(Just then, the white the) < P(Just then, the white rabbit)

- Contextual information limited to n-value (cfr. n-gram models)

Language Model

- P(sentence) = ∏P(word)
- Unigram: P(word) = freq(word) / N
- bigram: $P(word_i|word_{i-1})$ = freq('$word_{i-1}$ $word_i$') / freq($word_{i-1}$)

P(rabbit|white) = freq(white rabbit)/freq(white)

P(the|white) = freq(white the)/freq(the)

data driven: given $n_1, n_2, n_3, n_4, \ldots n_z$

Language Model

- $P(\text{sentence}) = \prod P(\text{word})$
- Unigram: $P(\text{word}) = \text{freq}(\text{word}) / N$
- bigram: $P(\text{word}_i | \text{word}_{i-1}) = \text{freq}(\text{`word}_{i-1}\ \text{word}_i\text{'}) / \text{freq}(\text{word}_{i-1})$

P(rabbit|white) = freq(white rabbit)/freq(white)

P(the|white) = freq(white the)/freq(the)

data driven: given $n_1, n_2, n_3, n_4, \ldots n_z$

Language Model

- $P(sentence) = \prod P(word)$
- Unigram: $P(word) = freq(word) / N$
- bigram: $P(word_i|word_{i-1}) = freq(\text{`}word_{i-1}\ word_i\text{'}) / freq(word_{i-1})$

P(rabbit|white) = freq(white rabbit)/freq(white)

P(the|white) = freq(white the)/freq(the)

data driven: given $n_1, n_2, n_3, n_4, \ldots n_z$

Language Model

- P(sentence) = ∏P(word)
- Unigram: P(word) = freq(word) / N
- bigram: $P(word_i|word_{i-1}) = freq(`word_{i-1}\ word_i') / freq(word_{i-1})$
- trigram: $P(word_i|word_{i-2}word_{i-1}) = freq(`word_{i-2}\ word_{i-1}\ word_i') / freq(word_{i-2}word_{i-1})$

P(rabbit|the white) = freq(the white rabbit)/freq(the white)

P(the|the white) = freq(the white the)/freq(the white)

data driven: given $n_1, n_2, n_3, n_4, \ldots n_z$

Universiteit Antwerpen

Language Model

- P(sentence) = ∏P(word)
- Unigram: P(word) = freq(word) / N
- bigram: $P(word_i|word_{i-1})$ = freq('$word_{i-1}$ $word_i$') / freq($word_{i-1}$)
- trigram: $P(word_i|word_{i-2}word_{i-1})$ = freq('$word_{i-2}$ $word_{i-1}$ $word_i$') / freq($word_{i-2}word_{i-1}$)

P(rabbit|the white) = freq(the white rabbit)/freq(the white)

P(the|the white) = freq(the white the)/freq(the white)

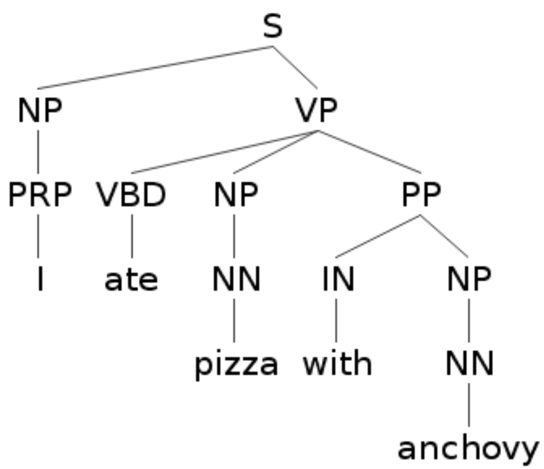
data driven: given $n_1, n_2, n_3, n_4, \ldots n_z$

N-gram models

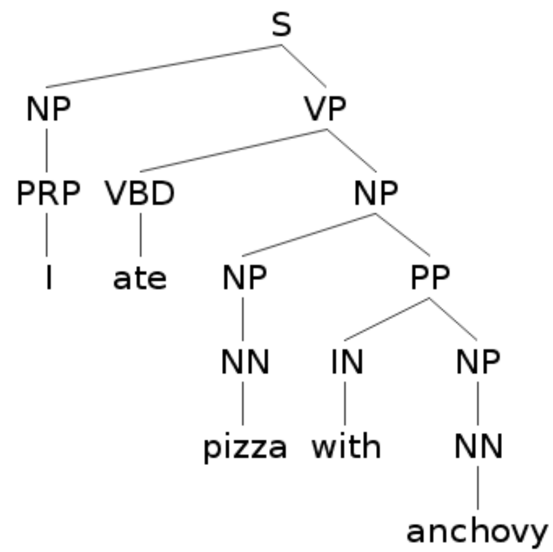
- The higher *n*, the more context is captured
- The higher *n*, the less statistical evidence we find for each context: sparse data problem

PCFG as a language model

$P(\text{tree}) = \prod P(\text{rule}_i)$



16/343



16/2401

$$P(\text{parse}) = \prod p(\text{rule}_i)$$
$$P(\text{sentence}) = \sum p(\text{parse}_k)$$
$$P(\text{text}) = \sum p(\text{sentence}_l)$$

Bayes' Rule & Noisy Channel

	P(Input)	P(Output|Input)
Machine Translation		Translation model
OCR		Model of OCR errors
Spellchecking	Language model	Model of spelling errors
POS-Tagging		Tag-Word Model
Speech Recognition		Acoustic model

Probabilistic Spelling Correction

Kernighan et al (1990): misspelt word differs from correct word in 1 substition, insertion, transposition or deletion

error	Correction	Correct Letter	Error Letter	Position	Type
acress	actress	t	-	2	deletion
acress	cress	-	a	0	insertion
acress	caress	ca	ac	0	transposition
acress	access	c	r	2	substitution
…					

- correction = argmaxP(t|c).P(c)

  with t=typo and C: list of correct words
- P(c): prior: language model (unigram)
- P(t|c): Model of misspellings

Probabilistic Spelling Correction

- Kernighan: 44x10^6 word AP newswire corpus
- PRIOR:

c	Freq(c)	P(c)
actress	1343	.0000315
cress	0	.000000014
caress	4	.0000001
access	2280	.000058

Probabilistic Spelling Correction

- Kernighan: 44×10^6 word AP newswire corpus
- PRIOR:

c	Freq(c)	P(c)
actress	1343	.0000315
cress	0	.000000014
caress	4	.0000001
access	2280	.000058

smoothing

Smoothing

- What if we want to calculate the probability of something we haven't seen yet?

  I like failblog

  + 'failblog' may not have been seen yet

  ➔ 0 probability

  ➔ 0 probability for entire sentence ($\prod P(word)$)

- Add-1 smoothing: $P(word) = \dfrac{freq(word) + \alpha}{N + \alpha.d}$

  with α: normalization factor (often $\alpha=1$)
  with N: total number of tokens (words)
  with d: total number of types (individual words)

Probabilistic Spelling Correction

- Model of misspellings: P(t|c)
- Proper P(t|c) cannot be computed, but can be estimated
- Use corpus of errors to construct confusion matrix of 26x26 for each type of mistake

del[x,y]: count how many times xy was typed as x

ins[x,y]: count how many times x was types as xy

sub[x,y]: count how many times x was typed as y

trans[x,y]: how many times xy was typed as yx

Probabilistic Spelling Correction

Correction	P(c)	P(t|c)	p(t|c)p(c)
actress	.0000315	.000117	3.69×10^{-9}
cress	.000000014	.00000144	2.02×10^{-14}
caress	.0000001	.0000164	1.64×10^{-13}
access	.000058	.000000209	1.21×10^{-11}

- acress is rewritten as ‘actress’
- use more intelligent prior to improve results in context

Bayes' Rule & Noisy Channel

	P(Input)	P(Output|Input)
Machine Translation		Translation model
OCR		Model of OCR errors
Spellchecking		Model of spelling errors
POS-Tagging	Language Model	Tag-Word Model
Speech Recognition		Acoustic model

Probabilistic n-gram POS Tagging

- Requires annotated corpus

  can/md the/dt tag/nn be/vb better/jjr

- Unigram: P(word|tag)P(tag)

  frequency of the tag for this word in corpus

- Bigram: $P(word_i|tag_i)\ P(tag_i|tag_{i-1})$

  frequency of the tag for this word in corpus, given previous tag

- Trigram: $P(word_i|tag_i)\ P(tag_i|tag_{i-1},tag_{i-2})$

  frequency of the tag for this word in corpus, given previous two tags

- Good Results, but possible data sparseness problems

Bayes' Rule & Noisy Channel

	P(Input)	P(Output|Input)
Machine Translation	Language Model	Translation model
OCR		Model of OCR errors
Spellchecking		Model of spelling errors
POS-Tagging		Tag-Word Model
Speech Recognition		Acoustic model

Modeling English Pronunciation Variation

- Differences in pronunciation
- 2 classes:
  - allophonic variation (due to context)

about	-	[ax b aw]	32%
	-	[ax b aw t]	16%
	-	[ix b aw]	8%

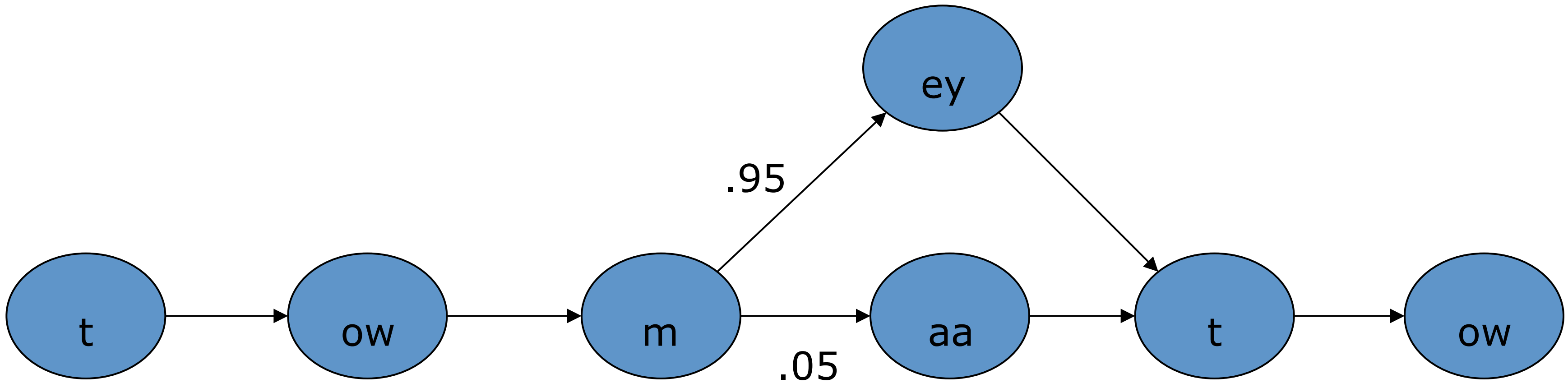
  - Lexical variation

about	-	[baw]	9%

Modeling English Pronunciation Variation

- we can model the distribution of this variation by introducing probabilities into a FSA

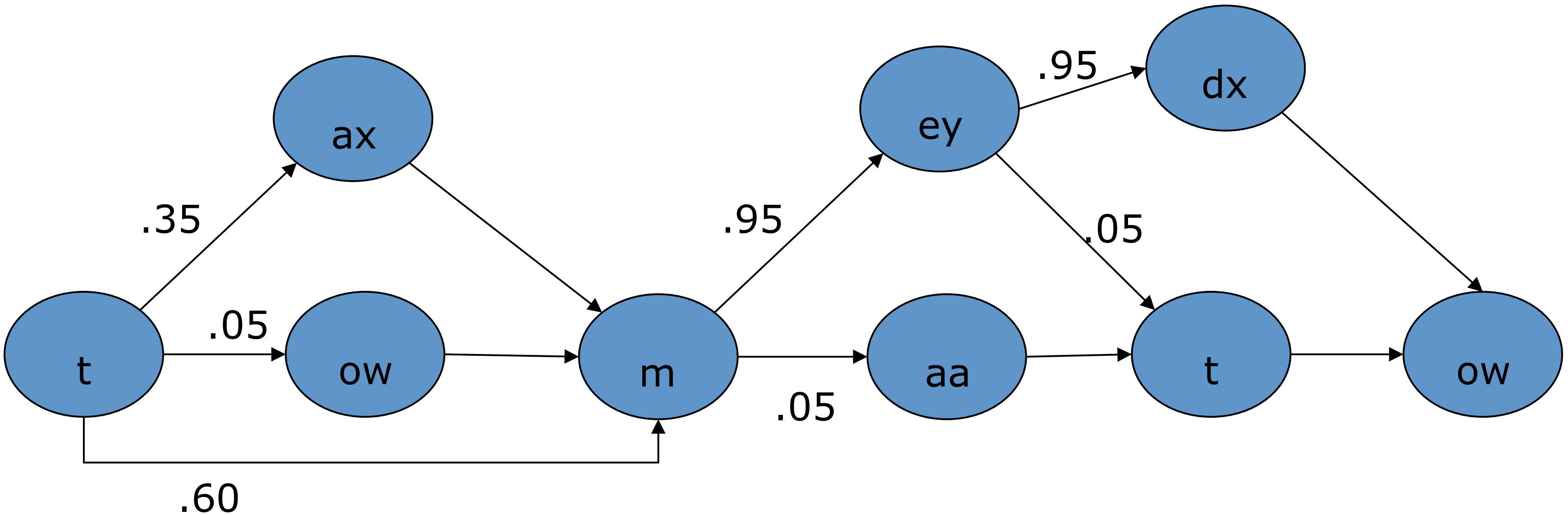
= a Weighted Automaton (Markov Chain)



- Models Sociolinguistic variation

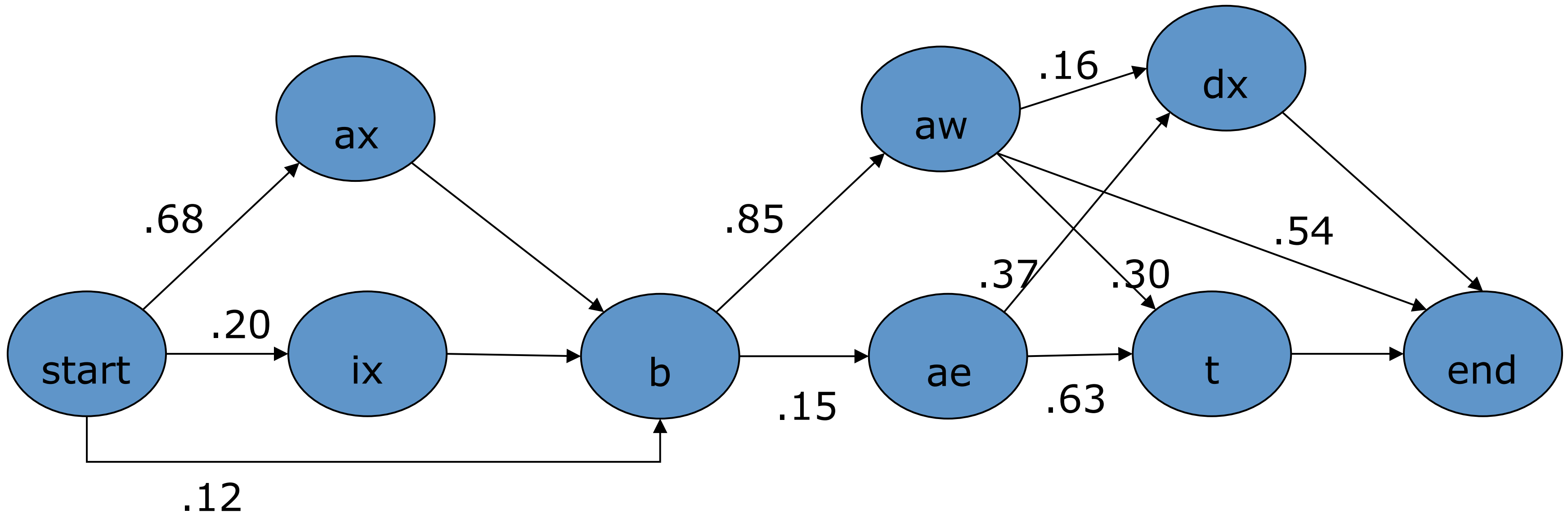
Modeling English Pronunciation Variation

- model allophonic variation:



Modeling English Pronunciation Variation

- "about": actual weighted automaton trained on pronunciations of Switchboard Corpus



Edit Distance

Minimum Edit Distance

- Spell checking: check writing against list of words/morphotactics
- Suggest list of alternatives?

  closest match

  fuzzy match
- *How to calculate the "distance" between two words:* minimum edit distance
- The minimal number of deletions, insertions, substitutions to go from word *a* to *b*

Minimum Edit Distance

- *intention* *vs* *execution*
- 3 operations (deletion, insertion, substitution)
- Alignment

i	n	t	e	*	n	t	i	o	n
*	e	x	e	c	u	t	i	o	n
d	s	s		i	s				

- Levenshtein distance: equal *weight* to all operations, no substitution (1substition = 1 deletion + 1 insertion)
- Levenshtein distance of *8* in example above

Minimum Edit Distance

	i n t e n t i o n
[delete i]	*n t e n t i o n*
[substitute n for e]	*e t e n t i o n*
[substitute t for x]	*e x e n t i o n*
[insert c]	*e x e c n t i o n*
[substitute n for u]	*e x e c u t i o n*

Minimum Edit Distance

- Computed through dynamic programming
- Solve problem by combining solutions to subproblems
- Table-driven

- Useful for
  - Alignment
  - Fuzzy string match
  - Spelling correction

Algorithm

Function LEVENSHTEIN-DISTANCE(*target*,*source*) returns *levenshtein-distance*

n ← length(*target*)

m ← length(*source*)

Create a distance matrix *distance[n+1,m+1]*

Initialize the 0th row and column to be the distance from the empty string

```
        distance[0,0] = 0
        for each column i from 1 to n do
                distance[i,0] ← distance[i-1,0] + 1 (= insertion cost)
        for each row j from 1 to m do
                distance[0,j] ← distance[0,j-1] + 1 (= deletion cost)
For each column i from 1 to n do
        for each row j from 1 to m do
                distance[i,j] ← MIN(    distance[i-1,j] + 1 (= insertion-cost),
                                        distance[i,j-1] + 1 (= deletion-cost),
                                        distance[i-1,j-1] + 2 (substition cost if A≠B)
                                        )
Return distance[n,m]
```

Edit distance matrix

n	*9*	↓8	↙←↓9	↙←↓10	↙←↓11	↙←↓12	↓11	↓10	↓9	**↙8**
o	*8*	↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↓10	↓9	**↙8**	↓9
i	*7*	↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↓9	**↙8**	←9	←10
t	*6*	↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	**↙8**	←9	←10	←↓11
n	*5*	↓4	↙←↓5	↙←↓6	↙←↓7	**↙←↓8**	↙←↓9	↙←↓10	↙←↓11	↙↓10
e	*4*	↙3	←4	**↙←↓5**	**←6**	←7	↙←↓8	↙←↓9	↙←↓10	↓9
t	*3*	↙←↓4	**↙←↓5**	↙←↓6	↙←↓7	↙←↓8	↙7	←↓8	↙←↓9	↓8
n	*2*	**↙←↓3**	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↓7	↙←↓8	↙7
i	***1***	↙←↓2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙6	←7	←8
ε	***0***	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
	ε	**e**	**x**	**e**	**c**	**u**	**t**	**i**	**o**	**n**

n	*9*	↓8	↙←↓9	↙←↓10	↙←↓11	↙←↓12	↓11	↓10	↓9	**↙8**
o	*8*	↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↓10	↓9	**↙8**	↓9
i	*7*	↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↓9	**↙8**	←9	←10
t	*6*	↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	**↙8**	←9	←10	←↓11
n	*5*	↓4	↙←↓5	↙←↓6	↙←↓7	**↙←↓8**	↙←↓9	↙←↓10	↙←↓11	↙↓10
e	*4*	↙3	←4	**↙←↓5**	**←6**	←7	↙←↓8	↙←↓9	↙←↓10	↓9
t	*3*	↙←↓4	**↙←↓5**	↙←↓6	↙←↓7	↙←↓8	↙7	←↓8	↙←↓9	↓8
n	*2*	**↙←↓3**	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↓7	↙←↓8	↙7
i	***1***	↙←↓2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙6	←7	←8
ε	***0***	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
	ε	**e**	**x**	**e**	**c**	**u**	**t**	**i**	**o**	**n**

Edit operations are determined by starting from the top right cell, following the arrows to find a path to cell0. Often, several paths are possible.

Path1

		i n t e n t i o n
COL0	*[delete i]*	*n t e n t i o n*
COL1	*[substitute n for e]*	*e t e n t i o n*
COL2	*[substitute t for x]*	*e x e n t i o n*
COL3		*e x e n t i o n*
COL4	*[insert c]*	*e x e c n t i o n*
COL5	*[substitute n for u]*	*e x e c u t i o n*
COL6		*e x e c u t i o n*
…COL9		*e x e c u t i o n*

Universiteit Antwerpen

n	*9*	↓8	↙←↓9	↙←↓10	↙←↓11	↙←↓12	↓11	↓10	↓9	**↙8**
o	*8*	↓7	↙←↓8	↙←↓9	↙←↓10	↙←↓11	↓10	↓9	**↙8**	↓9
i	*7*	↓6	↙←↓7	↙←↓8	↙←↓9	↙←↓10	↓9	**↙8**	←9	←10
t	*6*	↓5	↙←↓6	↙←↓7	↙←↓8	↙←↓9	**↙8**	←9	←10	←↓11
n	*5*	↓4	↙←↓5	**↙←↓6**	**↙←↓7**	**↙←↓8**	↙←↓9	↙←↓10	↙←↓11	↙↓10
e	*4*	**↙3**	**←4**	↙←↓5	←6	←7	↙←↓8	↙←↓9	↙←↓10	↓9
t	***3***	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↙7	←↓8	↙←↓9	↓8
n	***2***	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙←↓8	↓7	↙←↓8	↙7
i	***1***	↙←↓2	↙←↓3	↙←↓4	↙←↓5	↙←↓6	↙←↓7	↙6	←7	←8
ε	***0***	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
	ε	**e**	**x**	**e**	**c**	**u**	**t**	**i**	**o**	**n**

Edit operations are determined by starting from the top right cell, following the arrows to find a path to cell0. Often, several paths are possible.

Path1

		i n t e n t i o n
COL0a	*[delete i]*	*n t e n t i o n*
COL0b	*[delete n]*	*t e n t i o n*
COL0c	*[delete t]*	*e n t i o n*
COL1		*e n t i o n*
COL2	*[insert x]*	*e x n t i o n*
COL3	*[substitute n for e]*	*e x e t i o n*
COL4	*[insert c]*	*e x e c t i o n*
COL5	*[insert u]*	*e x e c u t i o n*

Exercise

Calculate the levenshtein distance between 'delen' en 'gedeeld'

n	*5*							
e	*4*							
l	*3*							
e	*2*							
d	*1*							
ε	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
	ε	**g**	**e**	**d**	**e**	**e**	**l**	**d**

Exercise

Calculate the levenshtein distance between

‘delen’ en ‘gedeeld’

n	*5*	↙←↓6						
e	*4*	↙←↓5						
l	*3*	↙←↓4						
e	*2*	↙←↓3						
d	*1*	↙←↓2						
ε	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
	ε	**g**	**e**	**d**	**e**	**e**	**l**	**d**

Exercise

Calculate the levenshtein distance between

'delen' en 'gedeeld'

n	*5*	↙←↓6	↓5					
e	*4*	↙←↓5	↙↓4					
l	*3*	↙←↓4	↓3					
e	*2*	↙←↓3	↙2					
d	*1*	↙←↓2	↙←↓3					
ε	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
	ε	**g**	**e**	**d**	**e**	**e**	**l**	**d**

Exercise

Calculate the levenshtein distance between
‘delen’ en ‘gedeeld’

n	*5*	↙←↓6	↓5	↙←↓6				
e	*4*	↙←↓5	↙↓4	↙←↓5				
l	*3*	↙←↓4	↓3	↙←↓4				
e	*2*	↙←↓3	↙2	←↓3				
d	*1*	↙←↓2	↙←↓3	↙2				
ε	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
	ε	**g**	**e**	**d**	**e**	**e**	**l**	**d**

Exercise

Calculate the levenshtein distance between 'delen' en 'gedeeld'

n	*5*	↙←↓6	↓5	↙←↓6	↓5			
e	*4*	↙←↓5	↙↓4	↙←↓5	↙↓4			
l	*3*	↙←↓4	↓3	↙←↓4	↓3			
e	*2*	↙←↓3	↙2	←↓3	↙2			
d	*1*	↙←↓2	↙←↓3	↙2	←3			
ε	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
	ε	**g**	**e**	**d**	**e**	**e**	**l**	**d**

Exercise

Calculate the levenshtein distance between 'delen' en 'gedeeld'

n	*5*	↙←↓6	↓5	↙←↓6	↓5	↓4		
e	*4*	↙←↓5	↙↓4	↙←↓5	↙↓4	↙3		
l	*3*	↙←↓4	↓3	↙←↓4	↓3	↙←↓4		
e	*2*	↙←↓3	↙2	←↓3	↙2	↙←3		
d	*1*	↙←↓2	↙←↓3	↙2	←3	←4		
ε	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
	ε	**g**	**e**	**d**	**e**	**e**	**l**	**d**

Exercise

Calculate the levenshtein distance between 'delen' en 'gedeeld'

n	*5*	↙←↓6	↓5	↙←↓6	↓5	↓4	↙←↓5	
e	*4*	↙←↓5	↙↓4	↙←↓5	↙↓4	↙3	←↓4	
l	*3*	↙←↓4	↓3	↙←↓4	↓3	↙←↓4	↙3	
e	*2*	↙←↓3	↙2	←↓3	↙2	↙←3	←4	
d	*1*	↙←↓2	↙←↓3	↙2	←3	←4	←5	
ε	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
	ε	**g**	**e**	**d**	**e**	**e**	**l**	**d**

Exercise

Calculate the levenshtein distance between
'delen' en 'gedeeld'

n	*5*	↙←↓6	↓5	↙←↓6	↓5	↓4	↙←↓5	↙←↓6
e	*4*	↙←↓5	↙↓4	↙←↓5	↙↓4	↙3	←↓4	↙←↓5
l	*3*	↙←↓4	↓3	↙←↓4	↓3	↙←↓4	↙3	←4
e	*2*	↙←↓3	↙2	←↓3	↙2	↙←3	←4	←5
d	*1*	↙←↓2	↙←↓3	↙2	←3	←4	←5	↙←6
ε	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
	ε	**g**	**e**	**d**	**e**	**e**	**l**	**d**

n	*5*	↙←↓6	↓5	↙←↓6	↓5	↓4	↙←↓5	**↙←↓6**
e	*4*	↙←↓5	↙↓4	↙←↓5	↙↓4	↙3	**←↓4**	↙←↓5
l	*3*	↙←↓4	↓3	↙←↓4	↓3	↙←↓4	**↙3**	←4
e	*2*	↙←↓3	↙2	←↓3	**↙2**	**↙←3**	←4	←5
d	*1*	↙←↓2	↙←↓3	**↙2**	←3	←4	←5	↙←6
ε	***0***	***1***	***2***	*3*	*4*	*5*	*6*	*7*
	ε	**g**	**e**	**d**	**e**	**e**	**l**	**d**

		d e l e n
COL1	*[insert g]*	*g d e l e n*
COL2	*[insert e]*	*g e d e l e n*
COL3		*g e d e l e n*
COL4		*g e d e l e n*
COL5	*[insert e]*	*g e d e e l e n*
COL6		***g*** *e d e e l e n*
COL6b	*[delete e]*	***g*** *e d e e l n*
COL7	*[substitute n for d]*	***g*** *e d e e l d*

Zie ook: http://www.let.rug.nl/kleiweg/lev/

Assignment

```
>>> from nltk.corpus import brown
>>> corpus = brown.sents()
>>> corpus[4]
[u'The', u'jury', u'said', u'it', u'did', u'find', u'that', u'many', u'of', u"Georgia's", u'registration', u'and', u'election', u'laws', u'``', u'are', u'outmoded', u'or', u'inadequate', u'and', u'often', u'ambiguous', u"''", u'.']
```

Write a script that extracts a trigram language model from this corpus. You can do this in 5 steps:

1. Create a dictionary (trigrams = {}) and add all trigrams in the corpus (key) and their associated count (value).
2. Create a dictionary (bigrams = {}) and add all bigrams in the corpus (key) and their associated count (value).
3. For every key in the trigram dictionary, divide the count by the value of the relevant bigram
4. Your trigram dictionary now contains probabilities
5. (save the dictionary using pickle)

Write a script that computes the probability of a sentence, according to your language model

```
>>> probability(corpus[4]) = <some value>
```

DEADLINE: 22 December 2014

Send python code through e-mail to guy.depauw@uantwerpen.be

Don’t hesitate to contact your helpline guy.depauw@uantwerpen.be