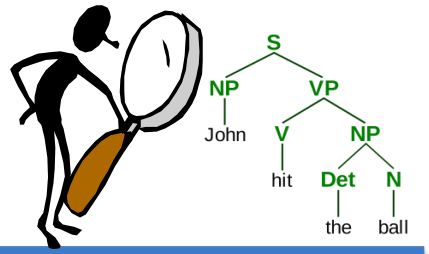


Wyszukiwanie i Przetwarzanie Informacji – NLP

Information Retrieval & Search

Irmina Masłowska

irmina.maslowska@cs.put.poznan.pl

www.cs.put.poznan.pl/imaslowska/wipi/

Natural Language Processing

- Przetwarzanie języka naturalnego (ang. *natural language processing*, NLP) to interdyscyplinarna dziedzina, łącząca zagadnienia sztucznej inteligencji i językoznawstwa, zajmująca się automatyzacją analizy, rozumienia, tłumaczenia i generowania języka naturalnego przez komputer
- Termin *język naturalny* używany jest, by odróżnić języki ludzkie (takie jak polski czy angielski) od języka formalnego czy komputerowego (jak C++, C#, Java lub Python)

Natural Language Processing

- Ze względu na niejednoznaczność praktycznie każdego dłuższego zdania w języku naturalnym w analizie języka naturalnego używa się metod statystycznych i opartych na rachunku prawdopodobieństwa
- W tym celu bazuje się na dużych korpusach tekstów w językach naturalnych
- Statystyczne NLP wywodzi się przede wszystkim z uczenia maszynowego i analizy danych

Probabilistic Language Models

- Cel: przypisanie prawdopodobieństwa do zdania lub frazy – sekwencji wyrazów

  $P(W) = P(w_1, w_2, w_3, \ldots, w_n)$

- Zadanie podobne: prawdopodobieństwo następnego wyrazu

  $P(w_n \mid w_1, w_2, w_3, \ldots, w_{n-1})$

- *Language model* – model, który dla każdej sekwencji wyrazów pozwala na wyznaczenie albo $P(W)$, albo $P(w_n \mid w_1, w_2, w_3, \ldots, w_{n-1})$

Computing P(W)

- Jak obliczyć prawdopodobieństwo łączne frazy?

  $P(W) = P(w_1, w_2, w_3, \ldots, w_n)$

  np. *woda w stawie była czysta jak łza*

Computing P(W) – The Chain Rule

- Jak obliczyć prawdopodobieństwo łączne frazy?

$P(W) = P(w_1, w_2, w_3, \ldots, w_n)$

Prawdopodobieństwo warunkowe:

$P(B|A) = P(A,B)/P(A) \Rightarrow P(A,B) = P(A)*P(B|A)$

Dla większej liczby zdarzeń:

$P(A,B,C,D) = P(A)*P(B|A)*P(C|A,B)*P(D|A,B,C)$

Reguła łańcuchowa:

$P(w_1,w_2,w_3, \ldots w_n) = P(w_1) * P(w_2|w_1) * P(w_3|w_1,w_2) * \ldots * P(w_n \mid w_1,w_2,w_3,\ldots w_{n-1})$

Computing P(W) – The Chain Rule

- Reguła łańcuchowa – wzór ogólny

$$P(w_1, w_2, w_3, \ldots w_n) = \prod_{i=1}^{n} P(w_i \mid w_1, w_2, w_3, \ldots w_{i-1})$$

$P(woda\ w\ stawie\ była\ czysta\ jak\ łza) =$

$P(woda)$
$* P(w \mid woda)$
$* P(stawie \mid woda\ w)$
$* P(była \mid woda\ w\ stawie)$
$* P(czysta \mid woda\ w\ stawie\ była)$
$* P(jak \mid woda\ w\ stawie\ była\ czysta)$
$* P(łza \mid woda\ w\ stawie\ była\ czysta\ jak)$

Estimating probabilities

$$P(w_1, w_2, w_3, \ldots w_n) = \prod_{i=1}^{n} P(w_i \mid w_1, w_2, w_3, \ldots w_{i-1})$$

- Ale jak wyznaczyć te prawdopodobieństwa – może po prostu policzyć wystąpienia w jakimś dużym korpusie danego języka i podzielić?

$$\mathrm{P}(\textit{łza} \mid \textit{Woda w stawie była czysta jak}) = \frac{\mathrm{count}(\textit{Woda w stawie była czysta jak łza})}{\mathrm{count}(\textit{Woda w stawie była czysta jak})}$$

Estimating probabilities

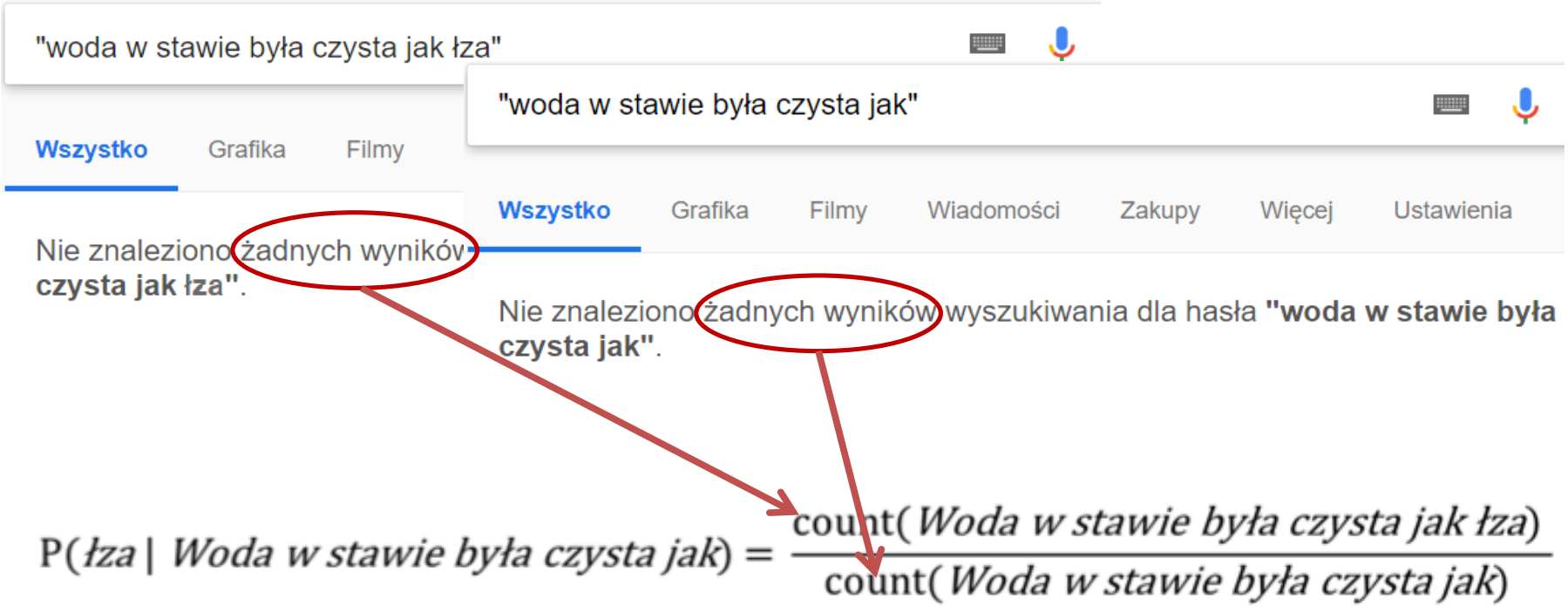
$$P(w_1, w_2, w_3, \ldots w_n) = \prod_{i=1}^{n} P(w_i \mid w_1, w_2, w_3, \ldots w_{i-1})$$

- Ale jak wyznaczyć te prawdopodobieństwa – może po prostu policzyć wystąpienia w jakimś dużym korpusie danego języka i podzielić?

$$\mathrm{P}(\textit{łza} \mid \textit{Woda w stawie była czysta jak}) = \frac{\mathrm{count}(\textit{Woda w stawie była czysta jak łza})}{\mathrm{count}(\textit{Woda w stawie była czysta jak})}$$

- Zbyt wiele potencjalnych zdań
- Nie wszystkie wystąpią nawet w największym korpusie

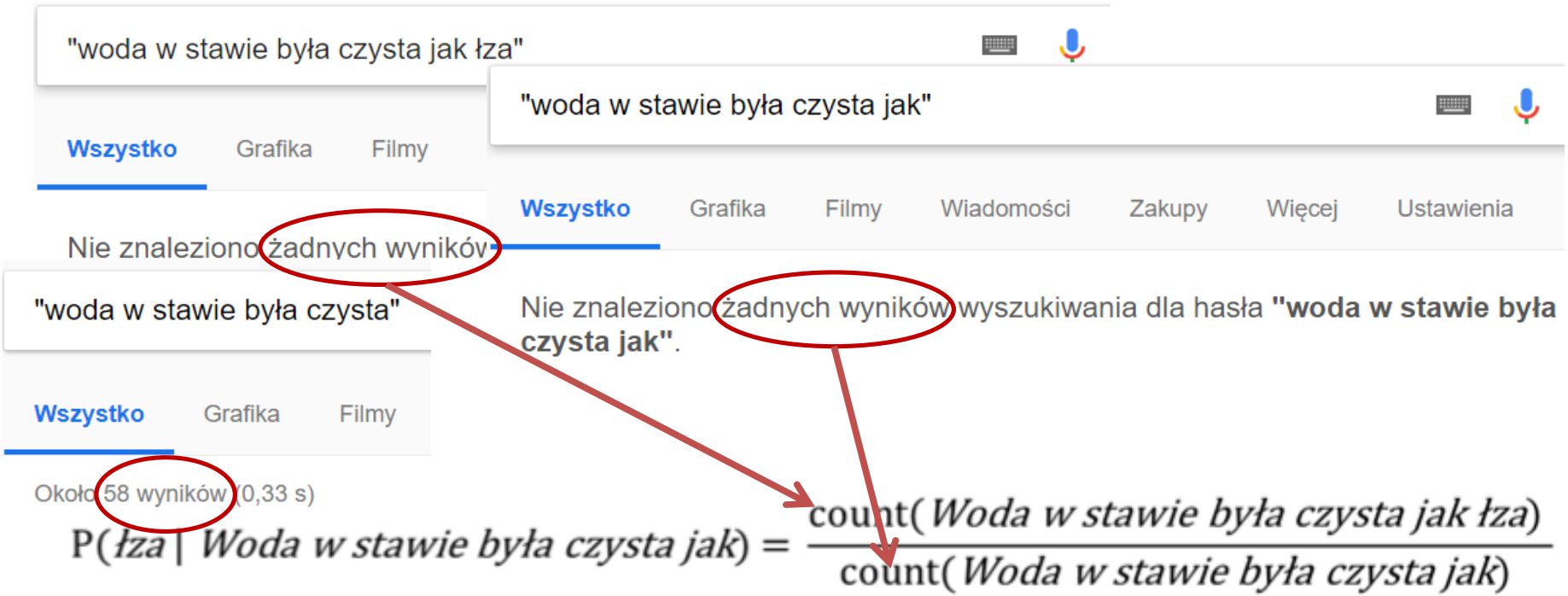
Estimating probabilities



$$P(\textit{łza} \mid \textit{Woda w stawie była czysta jak}) = \frac{\text{count}(\textit{Woda w stawie była czysta jak łza})}{\text{count}(\textit{Woda w stawie była czysta jak})}$$

- Zbyt wiele potencjalnych zdań
- Nie wszystkie wystąpią nawet w największym korpusie

Estimating probabilities



$$P(\textit{łza} \mid \textit{Woda w stawie była czysta jak}) = \frac{\text{count}(\textit{Woda w stawie była czysta jak łza})}{\text{count}(\textit{Woda w stawie była czysta jak})}$$

- Zbyt wiele potencjalnych zdań
- Nie wszystkie wystąpią nawet w największym korpusie

Markov assumption

- Własność Markowa

  Prawdopodobieństwo kolejnych zdarzeń zależy jedynie od bieżącego stanu:
  $P(łza|woda\ w\ stawie\ była\ czysta\ jak) \approx P(łza|jak)$

Markov assumption

- Własność Markowa

  Prawdopodobieństwo kolejnych zdarzeń zależy jedynie od bieżącego stanu:
  $P(łza|woda\ w\ stawie\ była\ czysta\ jak) \approx P(łza|jak)$

- Język naturalny w pewnym stopniu ma tę własność, ale może raczej:
  $P(łza|woda\ w\ stawie\ była\ czysta\ jak) \approx P(łza|czysta\ jak)$

- Przyjęcie podobnego założenia umożliwia estymację prawdopodobieństw dla wielu różnorodnych sekwencji i daje znaczne uproszczenie obliczeń

Language Modeling w/ N-grams

- Unigram – pojedynczy wyraz

$$P(w_1, w_2, w_3, \dots w_n) = \prod_{i=1}^{n} P(w_i)$$

- Bigram – dwa sąsiadujące wyrazy

$$P(w_1, w_2, w_3, \dots w_n) = \prod_{i=1}^{n} P(w_i \mid w_{i-1})$$

- Trigram – trzy sąsiadujące wyrazy

$$P(w_1, w_2, w_3, \dots w_n) = \prod_{i=1}^{n} P(w_i \mid w_{i-2}, w_{i-1})$$

- *N*-gram – uogólnienie: *N* sąsiadujących wyrazów

$$P(w_1, w_2, w_3, \dots w_n) = \prod_{i=1}^{n} P(w_i \mid w_{i-k}, \dots w_{i-2}, w_{i-1})$$

- **Estimating bigram probabilities**

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

Te wartości łatwo wyznaczyć:

– Liczba wystąpień bigramów

– Liczba wystąpień unigramów

- **Bigram example**

Zdania (frazy z wyróżnionym początkiem i końcem zdania):

<s>Idzie kominiarz po drabinie</s>

<s>Idzie kominiarz do domu po pracy</s>

<s>W domu i po pracy</s>

<s>Po pracy wróciłem do domu</s>

Prawdopodobieństwa:

$P(kominiarz|Idzie) =$ $P(Idzie|<s>) =$

$P(po|kominiarz) =$ $P(drabinie|po) =$

$P(pracy|po) =$

i trigram: $P(wróciłem|po\ pracy) =$

- **Bigram example**

  Zdania (frazy z wyróżnionym początkiem i końcem zdania):

  <s>Idzie kominiarz po drabinie</s>

  <s>Idzie kominiarz do domu po pracy</s>

  <s>W domu i po pracy</s>

  <s>Po pracy wróciłem do domu</s>

  Prawdopodobieństwa:

  $P(kominiarz|Idzie) = 2/2 \quad P(Idzie|<s>) = 2/4$

  $P(po|kominiarz) = 1/2 \quad P(drabinie|po) = 1/4$

  $P(pracy|po) = 3/4$

  i trigram: $P(wróciłem|po\ pracy) = 1/3$

Language Modeling w/ N-grams

- Kilka anglojęzycznych zdań automatycznie wygenerowanych na podstawie modelu *unigram*:

  „fifth, an, of, futures, the, an, incorporated, a, a, the, inflation, most, dollars, quarter, in, is, mass”

  „thrift, did, eighty, said, hard, 'm, july, bullish”

  „that, or, limited, the”

- Zdania dla modelu *bigram* (brany pod uwagę jest poprzedni wyraz):

  „texaco, rose, one, in, this, issue, is, pursuing, growth, in, a, boiler, house, said, mr., gurria, mexico, 's, motion, control, proposal, without, permission, from, five, hundred, fifty, five, yen”

  „outside, new, car, parking, lot, of, the, agreement, reached”

  „this, would, be, a, record, november”

Za: D. Jurafsky, J. Martin, *Speech and Language Processing (3rd ed. Draft)*

Language Modeling w/ N-grams

- Wykorzystanie trigramów, 4-gramów, 5-gramów mogłoby dać lepsze wyniki
- Jednak nawet te modele będą niewystarczające, gdyż język naturalny wykazuje także zależności obserwowalne dopiero dla bardzo dużych wartości *N*

  The computer(s) which I had just put into the machine room on the fifth floor is (are) crashing.
- W praktyce jednak modele *N*-gramowe wykorzystuje się często z powodzeniem

Za: D. Jurafsky, J. Martin, *Speech and Language Processing (3rd ed. Draft)*

Potencjalne zastosowania

- Tłumaczenie maszynowe:

High winds today – $P(Dzisiaj\ wysoki\ wiatr) < P(Dzisiaj\ silny\ wiatr)$

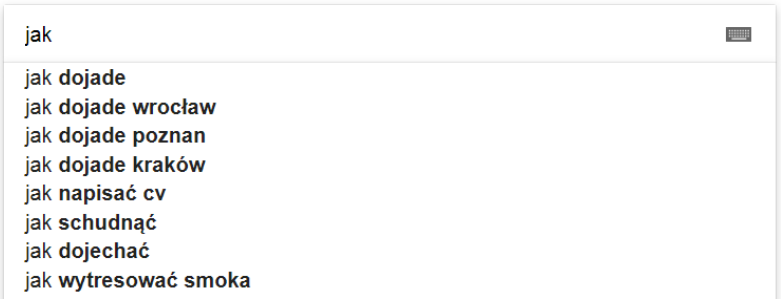
- Poprawianie błędów pisowni:

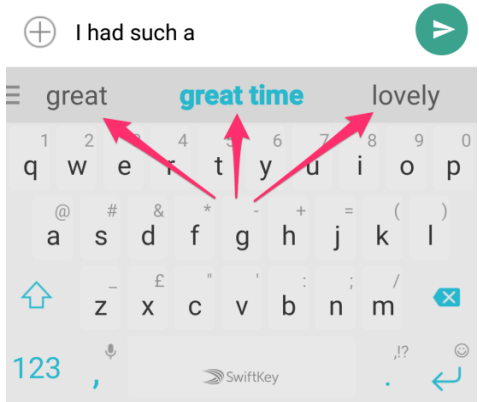
$P(Dzisiaj\ sliny\ wiatr) < P(Dzisiaj\ silny\ wiatr)$

- Rozpoznawanie mowy:

$P(Czy\ sąsiadki\ w\ sprzedaży?) < P(Czy\ są\ siatki\ w\ sprzedaży?)$

- Autouzupełnianie i podpowiedzi:





Example system – BeRP

Berkeley Restaurant Project

Obecnie nieczynny system dialogowy, który rozpoznawał mowę i udzielał informacji na temat różnych restauracji w Berkeley, Kalifornia (*Jurafsky i inni*, 1994)

https://youtu.be/d9gDcHBmr3I ; https://github.com/wooters/berp-trans

Przykładowe zapytania użytkowników

can you tell me about any good cantonese restaurants close by

mid priced thai food is what i'm looking for

tell me about chez panisse

can you give me a listing of the kinds of food that are available

i'm looking for a good place to eat breakfast

when is caffe venezia open during the day

Za: D. Jurafsky, J. Martin, *Speech and Language Processing (3rd ed. Draft)*

Example system – BeRP

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Przykład:

$P(want|i)$ = count(i,$want$) / count(i) = 827 / 2533 = 0,32649

Za: D. Jurafsky, J. Martin, *Speech and Language Processing (3rd ed. Draft)*

Example system – BeRP

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Przykład:

$P(want|i)$ = count(i,$want$) / count(i) = 827 / 2533 = 0,32649

Za: D. Jurafsky, J. Martin, *Speech and Language Processing (3rd ed. Draft)*

- **Bigram example – probability of a sentence**

Przykładowe zdanie:

I want Chinese food.

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

- **Bigram example – probability of a sentence**

Przykładowe zdanie:

I want Chinese food.

Prawdopodobieństwa:

P(<s> *i want chinese food* </s>)

= *P*(*i*|<s>) * *P*(*want*|*i*) * *P*(*chinese*|*want*)
* *P*(*food*|*chinese*) * (</s>|food)

= 0.25 * 0.33 * 0.0065 * 0.52 * 0.68

= 0.000189618

P(i|<s>) = 0.25

P(</s>|food) = 0.68

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Part-of-Speech tagging

Oznaczanie części mowy (POS-tagging) to jedno z klasycznych zadań w NLP – przetwarzaniu języka naturalnego – cieszące się ciągłą popularnością od kilku dekad

Inne określenia:

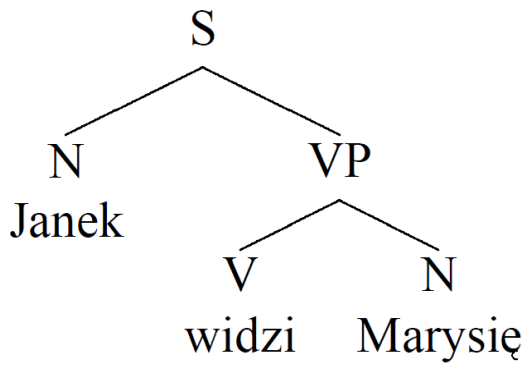
- *grammatical tagging*
- *word-category disambiguation*

Penn Treebank tagset

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two*	TO	"to"	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential 'there'	*there*	VB	verb base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb gerund	*eating*
JJ	adjective	*yellow*	VBN	verb past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, sing.	*IBM*	$	dollar sign	$
NNPS	proper noun, plural	*Carolinas*	#	pound sign	#
PDT	predeterminer	*all, both*	"	left quote	' or "
POS	possessive ending	*'s*	"	right quote	' or "
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one's*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

45-tag Penn Treebank tagset – za: D. Jurafsky, J. Martin, *Speech and Language Processing (3rd ed. Draft)*

Treebank – korpus tekstów w języku naturalnym, w którym każde zdanie poddano analize składniowej i oznaczono strukturą syntaktyczną Struktura zdania jest zazwyczaj reprezentowana w postaci drzewa



http://nlp.ipipan.waw.pl/CRIT2/

Ambiguity in English

book can be a verb (***book*** *that flight*) or a noun (*hand me that* ***book***)

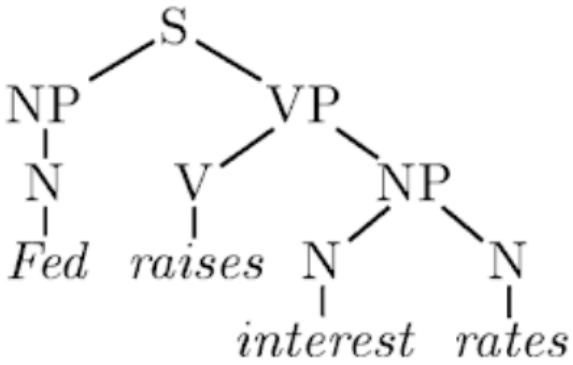
Some of the most ambiguous frequent words are: ***that***, ***back***, ***down***, ***put*** and ***set***

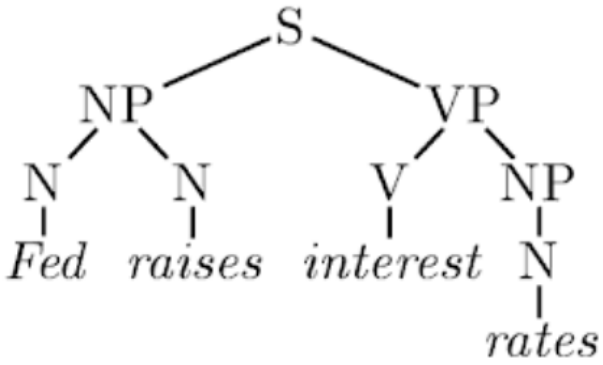
Examples of the 6 different parts-of-speech for the word ***back***:

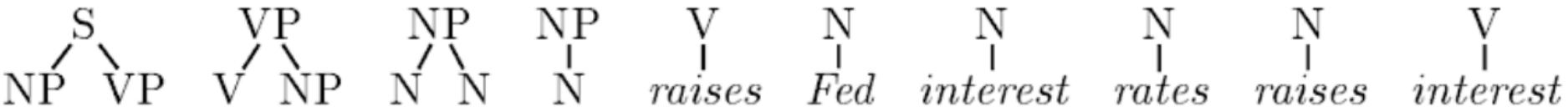
earnings growth took a ***back***/JJ *seat*
a small building in the ***back***/NN
a clear majority of senators ***back***/VBP *the bill*
Dave began to ***back***/VB *toward the door*
enable the country to buy ***back***/RP *about debt*
I was twenty-one ***back***/RB *then*

za: D. Jurafsky, J. Martin, *Speech and Language Processing (3rd ed. Draft)*

Ambiguity in English







Two syntactic analyses of an English sentence with a part-of-speech ambiguity that results in two very different structures

Ambiguity in English

Types:		WSJ	Brown
Unambiguous	(1 tag)	44,432 (**86%**)	45,799 (**85%**)
Ambiguous	(2+ tags)	7,025 (**14%**)	8,050 (**15%**)
Tokens:			
Unambiguous	(1 tag)	577,421 (**45%**)	384,349 (**33%**)
Ambiguous	(2+ tags)	711,780 (**55%**)	786,646 (**67%**)

The amount of tag ambiguity for word types in the Brown and WSJ corpora, from the Treebank-3 (45-tag) tagging. These statistics include punctuation as words, and assume words are kept in their original case

za: D. Jurafsky, J. Martin, *Speech and Language Processing (3rd ed. Draft)*

Ambiguity in English

Types:		WSJ	Brown
Unambiguous	(1 tag)	44,432 (**86%**)	45,799 (**85%**)
Ambiguous	(2+ tags)	7,025 (**14%**)	8,050 (**15%**)
Tokens:			
Unambiguous	(1 tag)	577,421 (**45%**)	384,349 (**33%**)
Ambiguous	(2+ tags)	711,780 (**55%**)	786,646 (**67%**)

The amount of tag ambiguity for word types in the Brown and WSJ corpora, from the Treebank-3 (45-tag) tagging. These statistics include punctuation as words, and assume words are kept in their original case

Prosta heurystyka tagowania słowa częścią mowy, która w zbiorze uczącym występowała dla tego słowa najczęściej (a w przypadku słów spoza zbioru uczącego tagiem NNP – proper noun) daje trafność ok. 90% - punkt odniesienia dla innych metod

za: D. Jurafsky, J. Martin, *Speech and Language Processing (3rd ed. Draft)*

Markov chain -> HMM

- Łańcuchy Markowa bazują na założeniu, że stan zależy wyłącznie od stanu poprzedniego
- Na nich z kolei bazują tzw. ukryte modele Markowa *hidden Markov model* (HMM):
  - w odróżnieniu do łańcuchów Markowa stany są ukryte (nie są obserwowalne)
  - na podstawie sekwencji obserwacji możemy odkryć, jakie stany ukryte wystąpiły w sekwencji
  - długość sekwencji obserwacji jest równa długości sekwencji stanów ukrytych – jeden do jeden

Ukryty model Markowa

- H – lista (n) stanów ukrytych
- O – lista (m) możliwych obserwacji
- π – rozkład (n) prawdopodobieństw stanu początkowego
- T – macierz (n x n) przejść pomiędzy stanami
- E – macierz (n x m) emisji obserwacji ze stanu

HMM – example

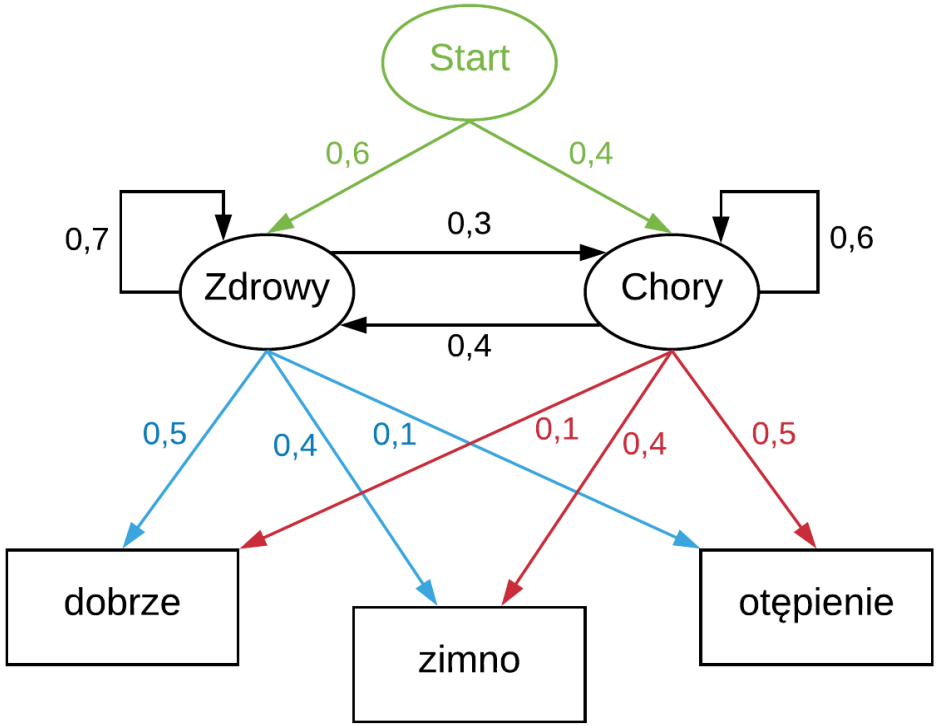
Rozpatrzmy internat, którego mieszkańcy mogą być *zdrowi* albo *chorzy*
Doktor diagnozuje ich raz dziennie pytając, jak się czują danego dnia
Możliwe są trzy odpowiedzi:

- *dobrze* (mieszkaniec czuje się dobrze)
- *zimno* (mieszkańcowi jest zimno)
- *otępienie* (mieszkaniec czuje się otępiały)

Doktor uważa, że stan mieszkańca w kolejnych dniach jest procesem Markowa i z uwagi na swoją wiedzę medyczną i doświadczenie zna prawdopodobieństwa: stanu początkowego, przejść oraz objawów

Wiedza doktora to ukryty model markowa

HMM – example



π rozkład stanu początkowego	**Zdrowy**	**Chory**
START	0,6	0,4

T macierz przejść	**Zdrowy**	**Chory**
Zdrowy	0,7	0,3
Chory	0,4	0,6

E macierz emisji	**dobrze**	**zimno**	**otępienie**
Zdrowy	0,5	0,4	0,1
Chory	0,1	0,4	0,5

The Viterbi Algorithm

function VITERBI(*o: observations* of len T, *s: state-graph* of len N) **returns** *best-path*

create a path probability matrix $viterbi[N+1,T]$
for each state s **from** 1 **to** N **do** ; initialization step
$viterbi[s,1] \leftarrow \pi[s] * \text{Emiss}[s,o[1]]$
$backpointer[s,1] \leftarrow 0$

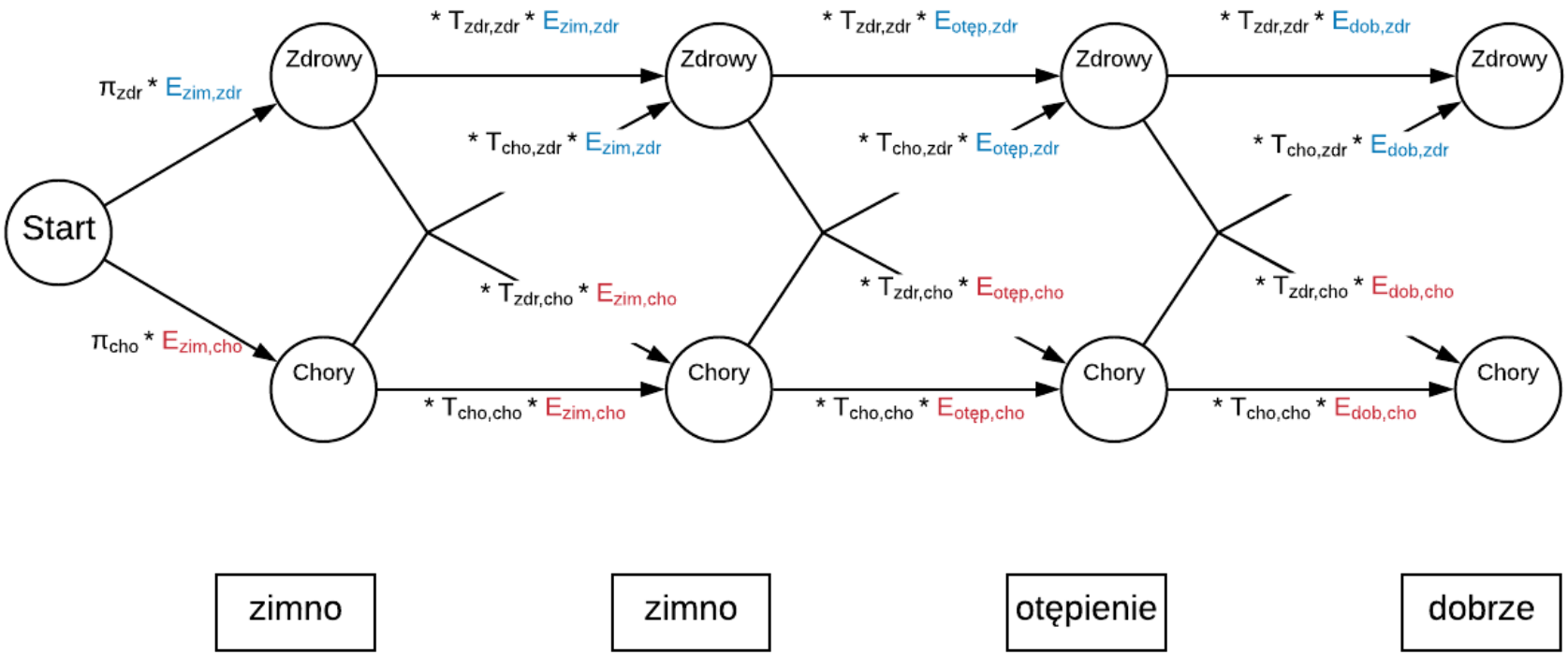
for each time step t **from** 2 **to** T **do** ; recursion step
for each state s **from** 1 **to** N **do**
$viterbi[s,t] \leftarrow \max_{s'=1..N}(viterbi[s',t-1] * \text{Trans}[s',s]) * \text{Emiss}[s,o[t]]$
$backpointer[s,o] \leftarrow \arg\max_{s'=1..N}(viterbi[s',t-1] * \text{Trans}[s',s])$

$viterbi[q_F,T] \leftarrow \max_{s'=1..N}(viterbi[s',T])$; termination step
$backpointer[q_F,T] \leftarrow \arg\max_{s'=1..N}(viterbi[s',T])$

return the backtrace path by following backpointers to states back in time from
$backpointer[q_F,T]$

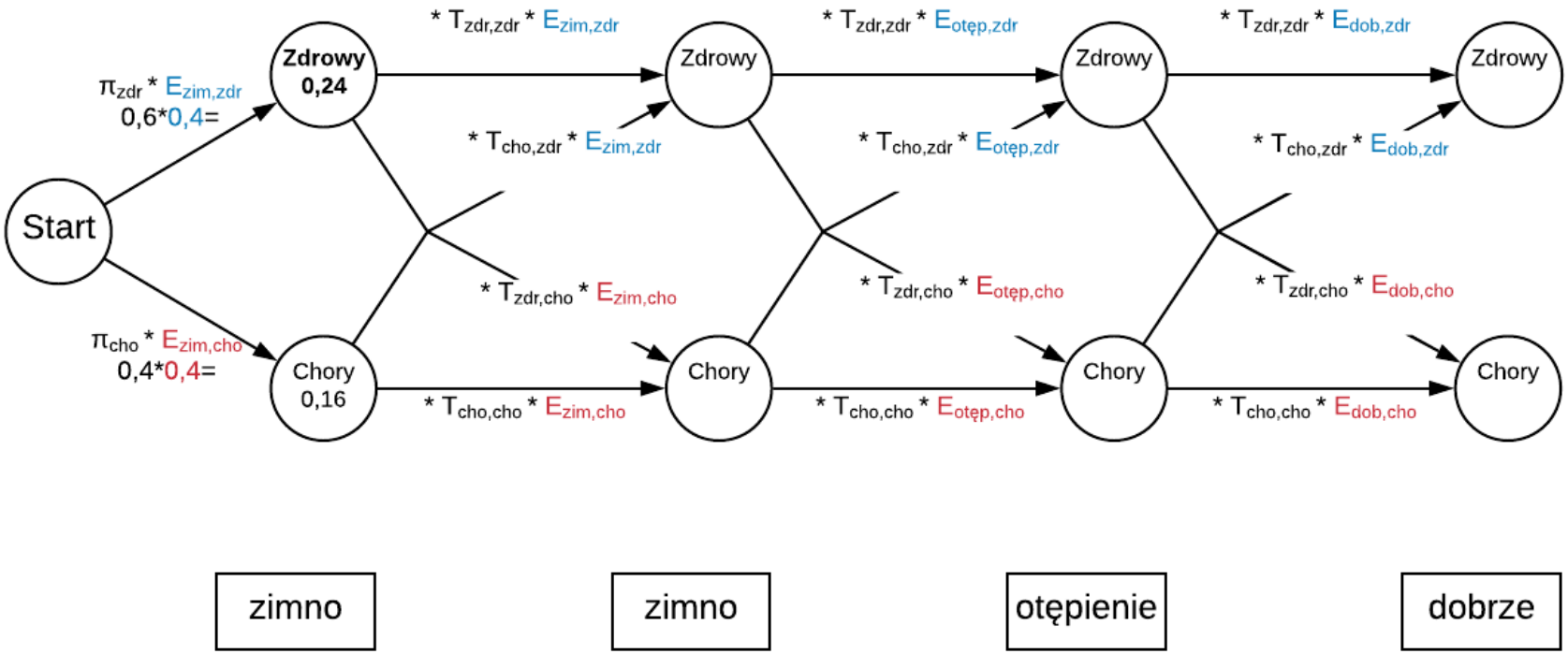
za: D. Jurafsky, J. Martin, *Speech and Language Processing (3rd ed. Draft)*

Viterbi – example



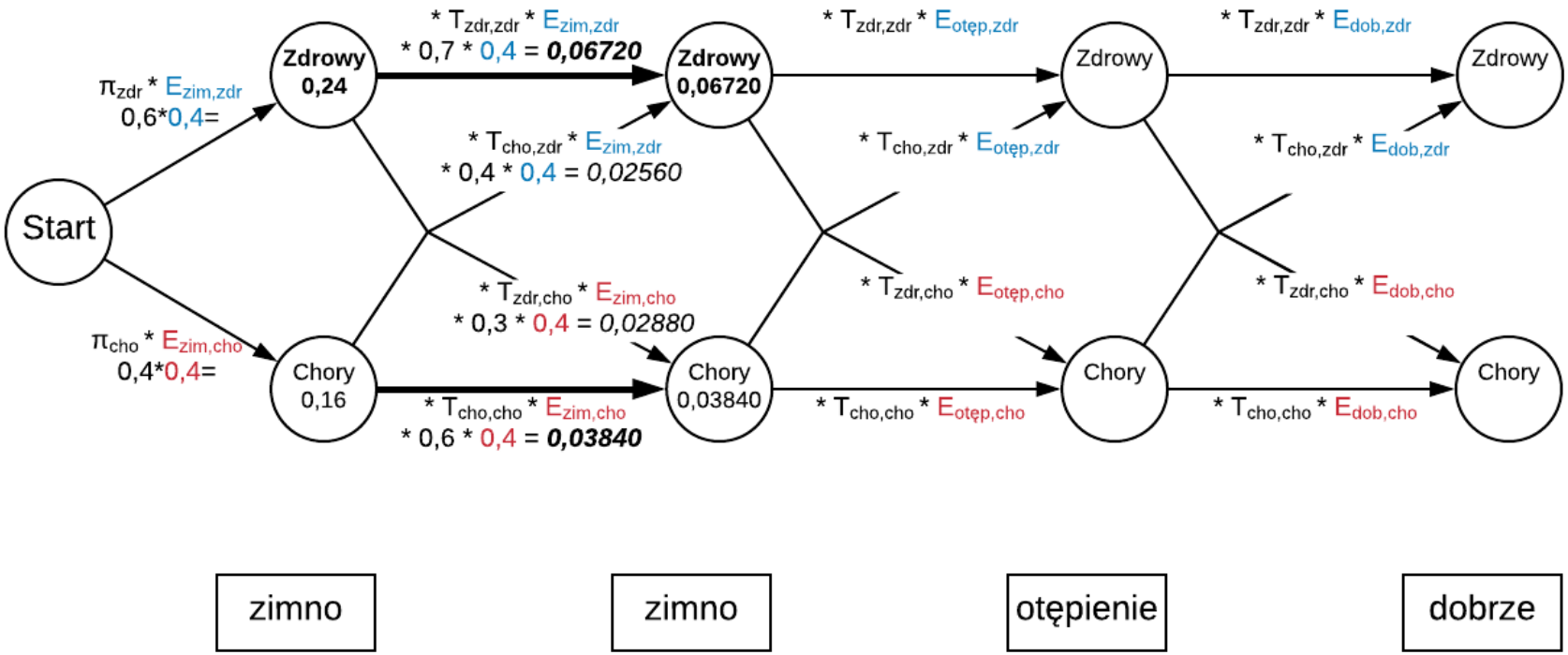
Wykres kratowy – *trellis*

Viterbi – example



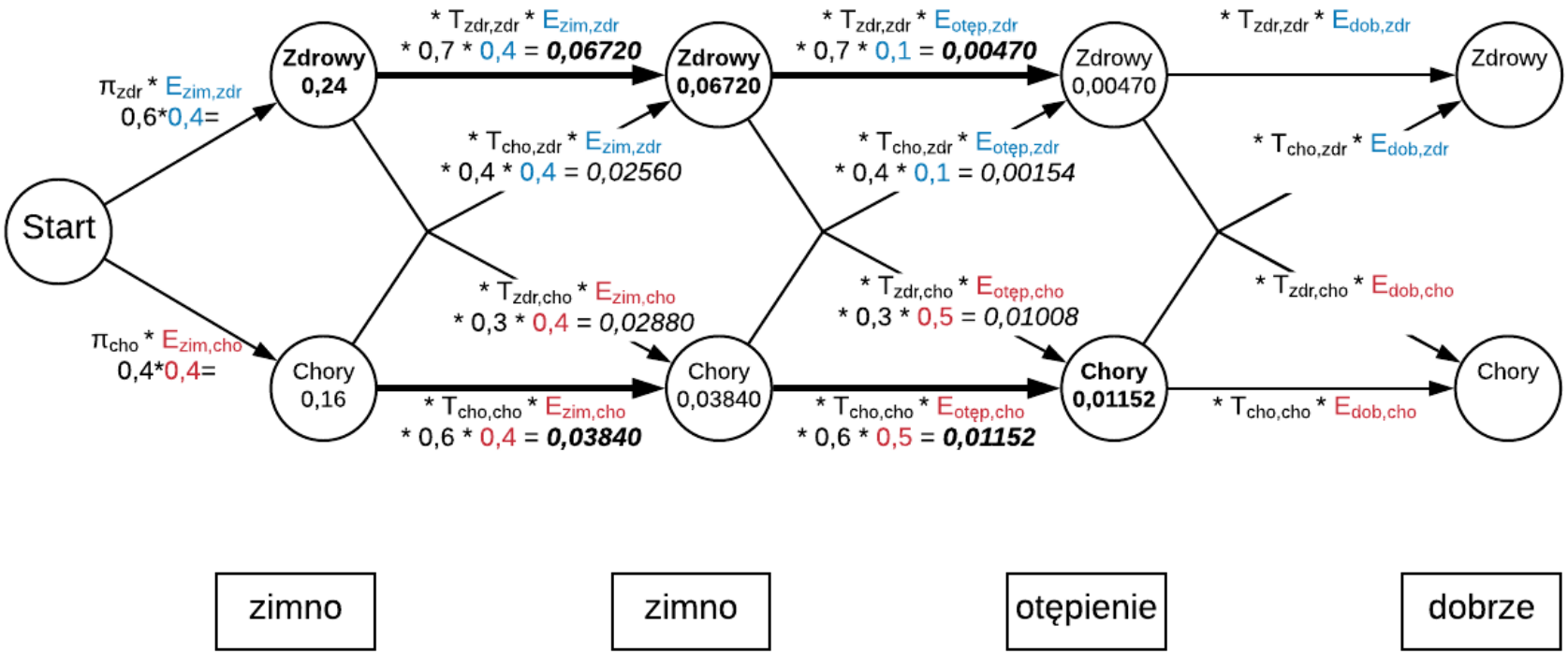
Wykres kratowy – *trellis*

Viterbi – example



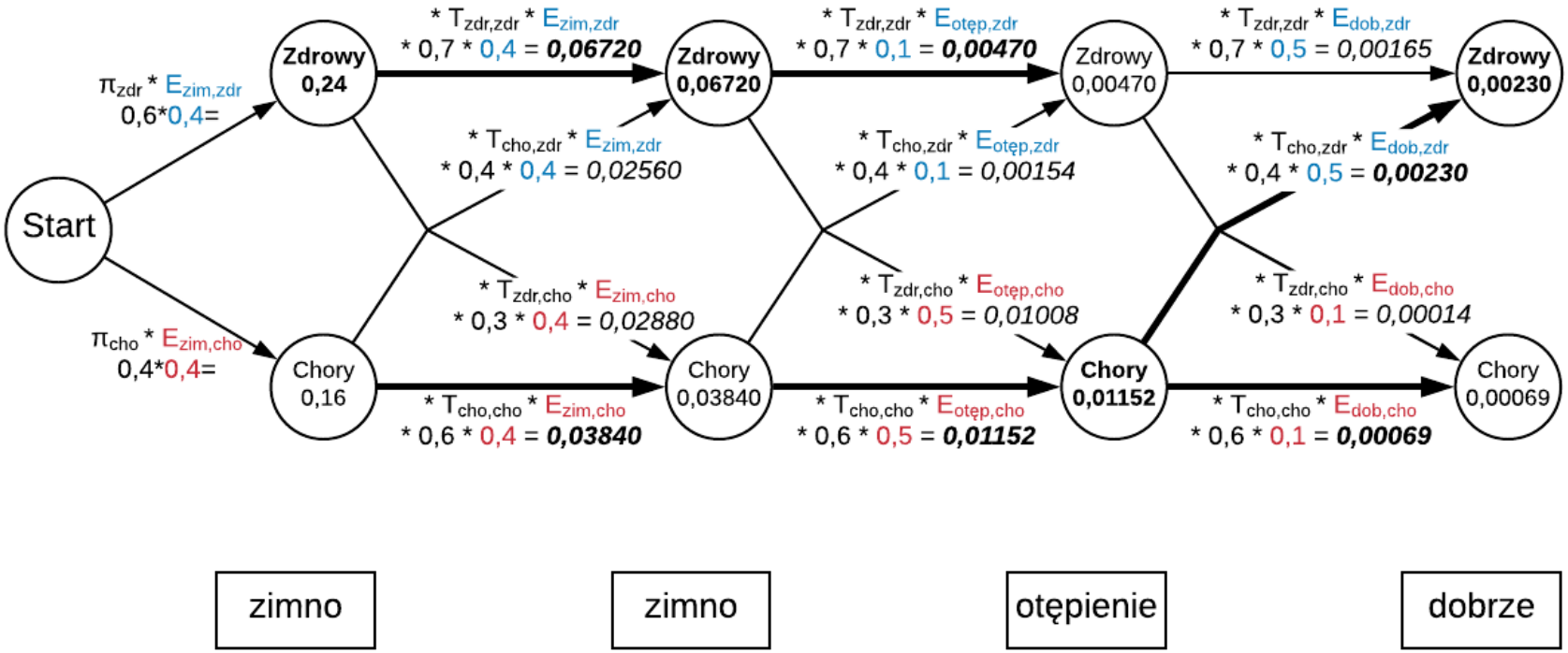
Wykres kratowy – *trellis*

Viterbi – example



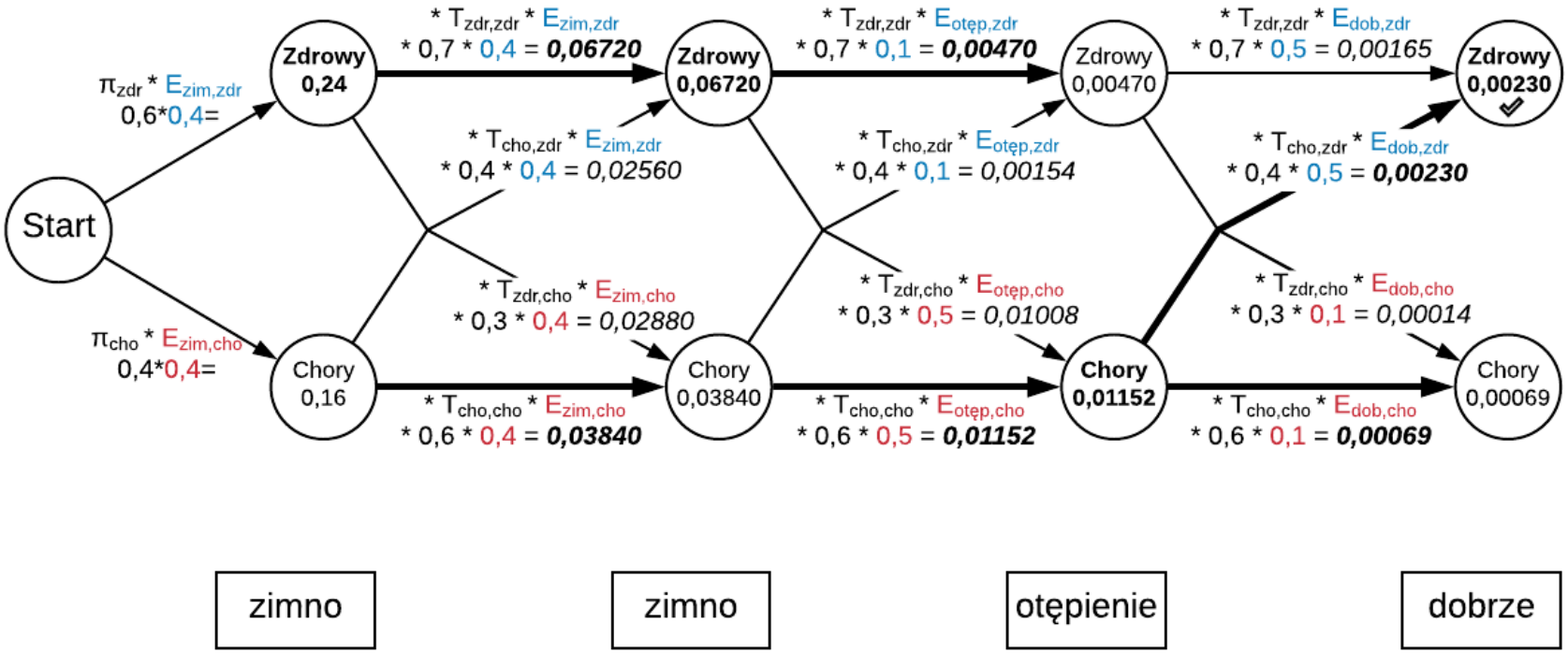
Wykres kratowy – *trellis*

Viterbi – example



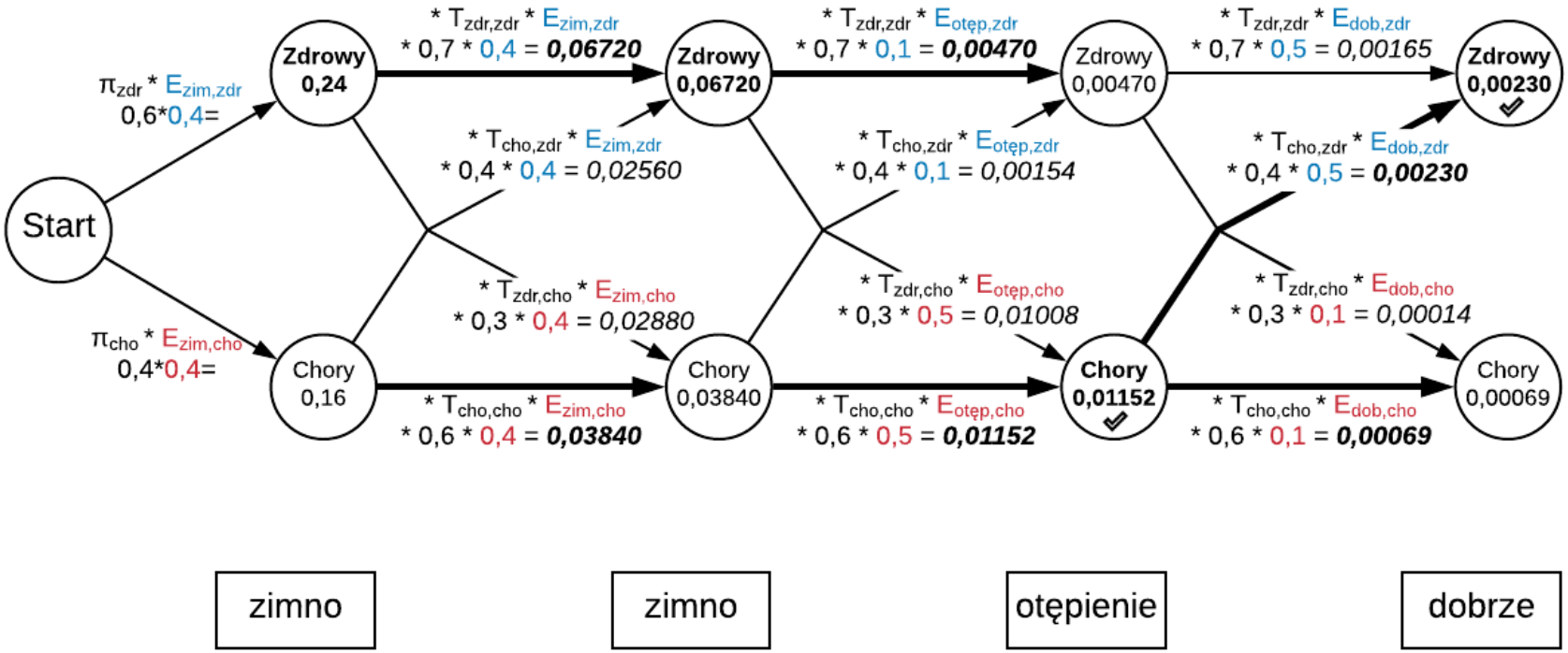
Wykres kratowy – *trellis*

Viterbi – example



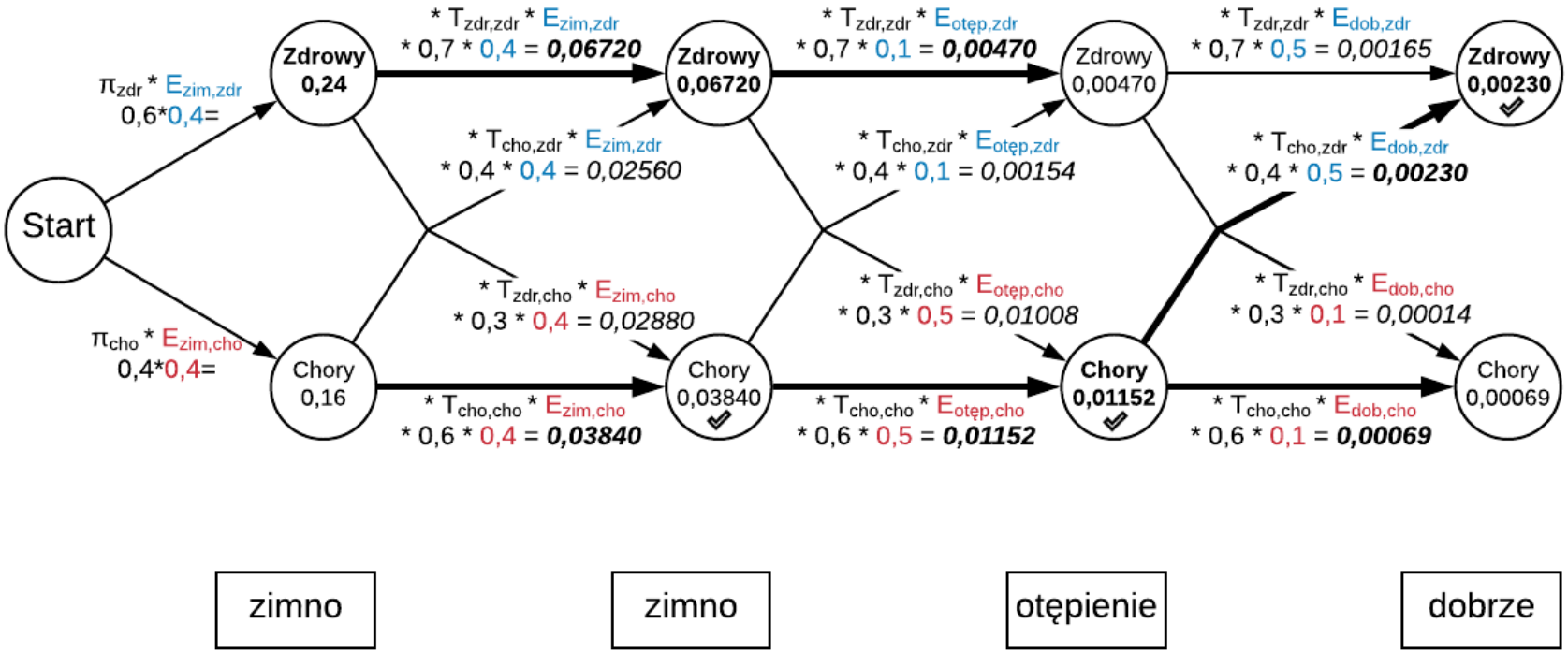
Wykres kratowy – *trellis*

Viterbi – example



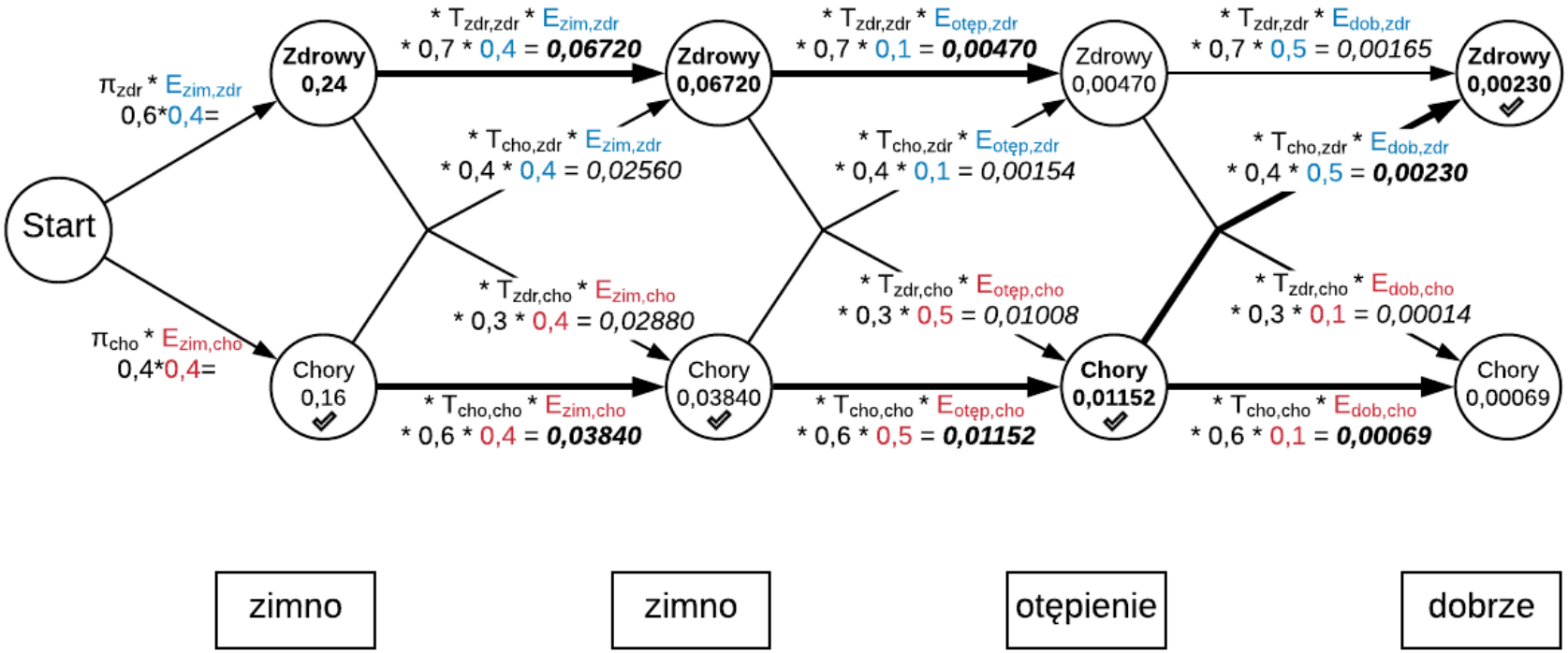
Wykres kratowy – *trellis*

Viterbi – example



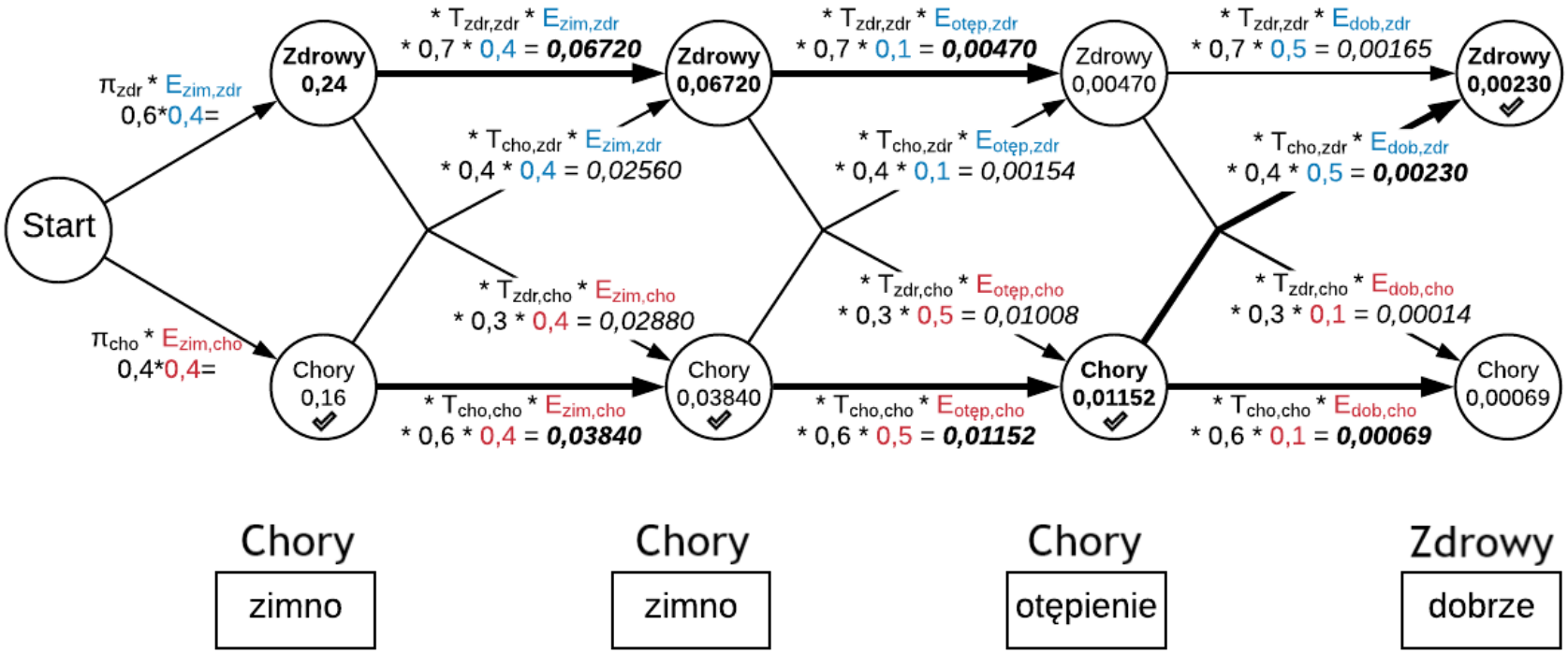
Wykres kratowy – *trellis*

Viterbi – example



Wykres kratowy – *trellis*

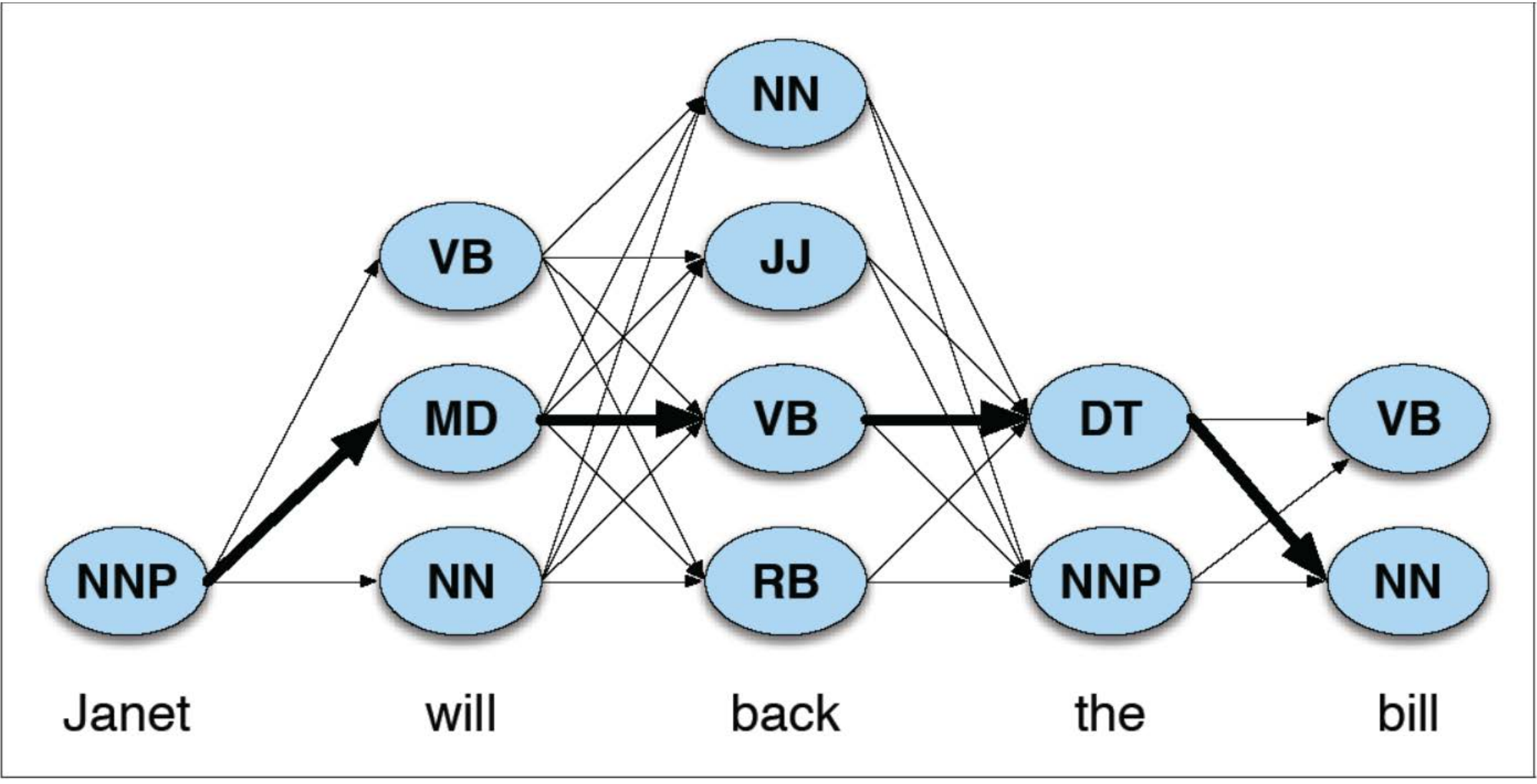
Viterbi – solution



Wykres kratowy – *trellis*

Ukryty model Markowa dla POS-tagging

- Stany ukryte odpowiadają częściom mowy
- Obserwacje to słowa tworzące strumień tekstu
- Część mowy (tag) słowa w_i zależy jedynie od części mowy słowa poprzedniego w_{i-1}
- Słowo jest generowane przez swoją część mowy, co oznacza, że jest zależne tylko od niej (a nie np. od poprzedniego słowa)


NN
VB
JJ
MD
VB
DT
VB
NNP
NN
RB
NNP
NN
Janet
will
back
the
bill

Exercises

I. Dana jest następująca kolekcja ucząca zdań:

<s>ja i ja</s> <s>ty to ja</s> <s>ja to ty</s> <s>ja to ty i to</s>

Jaki jest najbardziej prawdopodobny następny wyraz po wyrazie „to” a) wg modelu unigram, b) wg modelu bigram? Przyjmij P(ja)=5/14, P(i)=2/14, ...

II. W oparciu o powyższą kolekcję uczącą i model bigram oblicz prawdopodobieństwo zdania <s>ty i to ja</s>

III. Dla poniższego HMM i sekwencji obserwacji wylicz najbardziej prawdopodobną sekw. stanów ukrytych

H={X, Y}; O={a, b};

sekw=(a, a, b);

$$\pi = \begin{matrix} 0{,}5 & 0{,}5 \end{matrix} \qquad T = \begin{matrix} 0{,}8 & 0{,}2 \\ 0{,}4 & 0{,}6 \end{matrix} \qquad E = \begin{matrix} 0{,}8 & 0{,}2 \\ 0{,}4 & 0{,}6 \end{matrix}$$