

Wyszukiwanie i Przetwarzanie Informacji – NLP

Information Retrieval & Search

Irmina Maśłowska

irmina.maslowska@cs.put.poznan.pl

- 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)

- 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

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

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

- Zadanie podobne: prawdopodobieństwo następnego wyrazu

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

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

- Jak obliczyć prawdopodobieństwo łączne frazy?

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

np. *woda w stawie była czysta jak łza*

- Jak obliczyć prawdopodobieństwo łączne frazy?

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

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:

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

- Reguła łańcuchowa – wzór ogólny

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

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

$P(\text{woda})$

$* P(w \mid \text{woda})$

$* P(\text{stawie} \mid \text{woda w})$

$* P(\text{była} \mid \text{woda w stawie})$

$* P(\text{czysta} \mid \text{woda w stawie była})$

$* P(\text{jak} \mid \text{woda w stawie była czysta})$

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

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

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

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

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

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

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

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

Estimating probabilities

"woda w stawie była czysta jak łą"

Wszystko Grafika Filmy

Nie znaleziono żadnych wyników dla hasła "woda w stawie była czysta jak łą".

"woda w stawie była czysta jak"

Wszystko Grafika Filmy Wiadomości Zakupy Więcej Ustawienia

Nie znaleziono żadnych wyników wyszukiwania dla hasła "woda w stawie była czysta jak".

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

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

Estimating probabilities

"woda w stawie była czysta jak łąza"

Wszystko Grafika Filmy

Nie znaleziono żadnych wyników

"woda w stawie była czysta"

Wszystko Grafika Filmy

Około 58 wyników (0,33 s)

Nie znaleziono żadnych wyników wyszukiwania dla hasła "woda w stawie była czysta jak".

Wszystko Grafika Filmy Wiadomości Zakupy Więcej Ustawienia

$$P(łąza | Woda w stawie była czysta jak) = \frac{\text{count}(Woda w stawie była czysta jak łąza)}{\text{count}(Woda w stawie była czysta jak)}$$

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

- Własność Markowa

Prawdopodobieństwo kolejnych zdarzeń zależy jedynie od bieżącego stanu:

$$P(\textit{łza} | \textit{woda w stawie była czysta jak}) \approx P(\textit{łza} | \textit{jak})$$

- Własność Markowa

Prawdopodobieństwo kolejnych zdarzeń zależy jedynie od bieżącego stanu:

$$P(\textit{łza}|\textit{woda w stawie była czysta jak}) \approx P(\textit{łza}|\textit{jak})$$

- Język naturalny w pewnym stopniu ma tę własność, ale może raczej:

$$P(\textit{łza}|\textit{woda w stawie była czysta jak}) \approx P(\textit{łza}|\textit{czysta jak})$$

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

- Unigram – pojedynczy wyraz

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

- Bigram – dwa sąsiadujące wyrazy

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

- Trigram – trzy sąsiadujące wyrazy

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

- N-gram – uogólnienie: N sąsiadujących wyrazów

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

- **Estimating bigram probabilities**

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

Te wartości łatwo wyznaczyć:

- Liczba wystąpień bigramów
- Liczba wystąpień unigramów

■ Bigram example

Zdania (frazy z określonym początkiem i końcem zdania):

<s>Idzie kominiarz po drabinie</s>

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

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

<s>Po pracy idzie do domu</s>

Prawdopodobieństwa:

$$P(\text{kominiarz}|\text{idzie}) = P(\text{idzie}|\text{<s>}) =$$

$$P(\text{po}|\text{kominiarz}) = P(\text{drabinie}|\text{po}) =$$

$$P(\text{pracy}|\text{w}) = P(\text{</s>}|\text{pracy}) =$$

$$\dots \text{ i trigram: } P(\text{idzie}|\text{po pracy}) =$$

■ Bigram example

Zdania (frazy z określonym początkiem i końcem zdania):

<s>Idzie kominiarz po drabinie</s>

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

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

<s>Po pracy idzie do domu</s>

Prawdopodobieństwa:

$$P(\text{kominiarz}|\text{idzie}) = 2/3 \quad P(\text{idzie}|\text{<s>}) = 2/4$$

$$P(\text{po}|\text{kominiarz}) = 1/2 \quad P(\text{drabinie}|\text{po}) = 1/3$$

$$P(\text{pracy}|\text{w}) = 1/2 \quad P(\text{</s>}|\text{pracy}) = 2/3$$

$$\dots \text{ i trigram: } P(\text{idzie}|\text{po pracy}) = 1/2$$

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)

- 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 często wykorzystywane były z powodzeniem

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

Potencjalne zastosowania

- Tłumaczenie maszynowe:

High winds today – $P(\text{Dzisiaj } \textit{wysoki} \text{ wiatr}) < P(\text{Dzisiaj } \textit{silny} \text{ wiatr})$

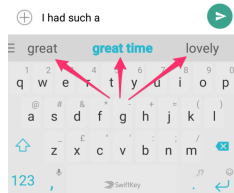
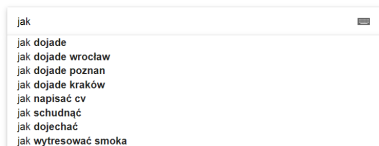
- Poprawianie błędów pisowni:

$P(\text{Dzisiaj } \textit{sliny} \text{ wiatr}) < P(\text{Dzisiaj } \textit{silny} \text{ wiatr})$

- Rozpoznawanie mowy:

$P(\text{Czy } \textit{sąsiadki} \text{ w sprzedaży?}) < P(\text{Czy } \textit{są siatki} \text{ w sprzedaży?})$

- Autouzupełnianie i podpowiedzi:



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:

$$P(\text{want}|i) = \text{count}(i, \text{want}) / \text{count}(i) = 827 / 2533 = 0,32649$$

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:

$$P(want|i) = \text{count}(i, \text{want}) / \text{count}(i) = 827 / 2533 = 0,32649$$

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:

$$\begin{aligned} &P(<s> \text{ i want chinese food } </s>) \\ &= P(i|<s>) * P(want|i) * P(chinese|want) \\ &\quad * P(food|chinese) * P(</s>|food) \\ &= 0.25 * 0.33 * 0.0065 * 0.52 * 0.68 \\ &= 0.000189618 \end{aligned}$$

Dane brakujące w tabeli:

$$P(i|<s>) = 0.25$$

$$P(</s>|food) = 0.68$$

	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

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

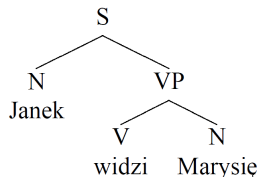
Inne określenia:

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

Penn Treebank tagset

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

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



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

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

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*

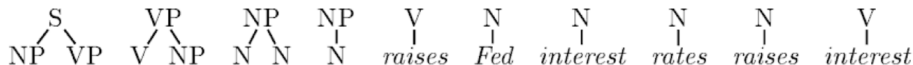
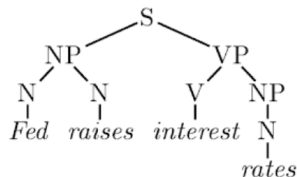
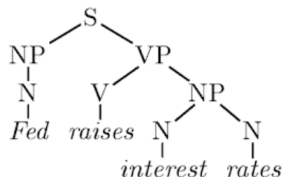
*Dave began to **back**/VB toward the door*

*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

Philosophy of Linguistics

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)*

- 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 ($n \times n$) przejść pomiędzy stanami
- E – macierz ($n \times m$) emisji obserwacji ze stanu

Rozpatrzmy internat, którego mieszkańcy mogą być *zdrowi* albo *chorzy*

Doktor diagnozuje ich raz dziennie pytając, jak się czują danego dnia

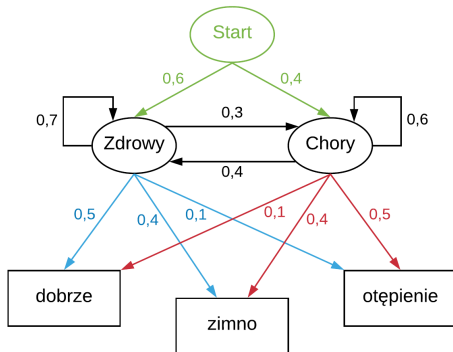
Możliwe są trzy odpowiedzi:

- *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



$$\pi$$

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

$viterbi[s, 1] \leftarrow \pi[s] * Emiss[s, o[1]]$

$backpointer[s, 1] \leftarrow 0$

for each time step *t* **from** 2 **to** *T* **do**

; recursion step

for each state *s* **from** 1 **to** *N* **do**

$viterbi[s, t] \leftarrow \max_{s'=1..N}(viterbi[s', t-1] * Trans[s', s]) * Emiss[s, o[t]]$

$backpointer[s, o] \leftarrow \arg \max_{s'=1..N}(viterbi[s', t-1] * Trans[s', s])$

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

; termination step

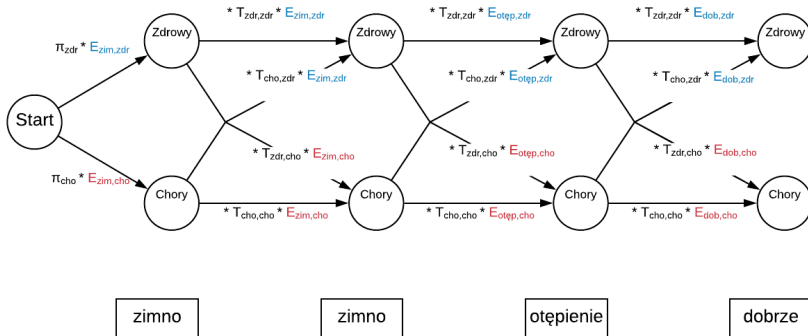
$backpointer[q_F, T] \leftarrow \arg \max_{s'=1..N}(viterbi[s', T])$

return the backtrace path by following backpointers to states back in time from

$backpointer[q_F, T]$

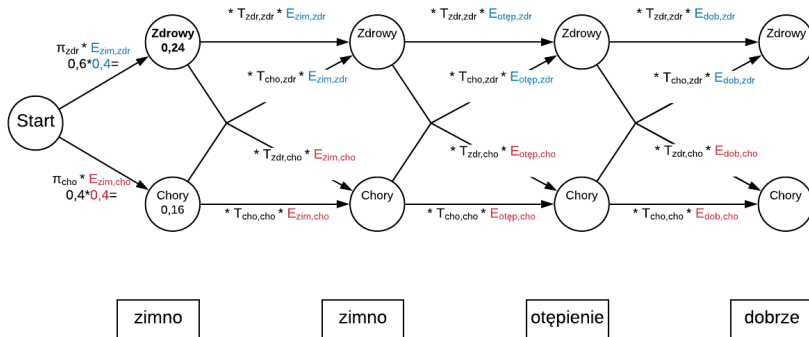
za: D. Jurafsky, J. Martin, *Speech and Language Processing (3rd ed. Draft)*

Viterbi – example



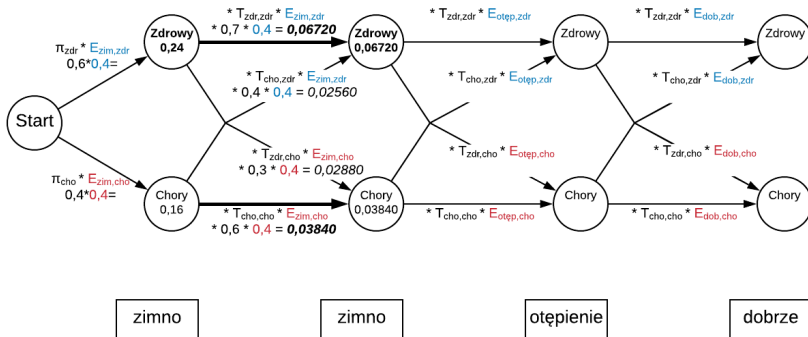
Wykres kratowy – *trellis*

Viterbi – example



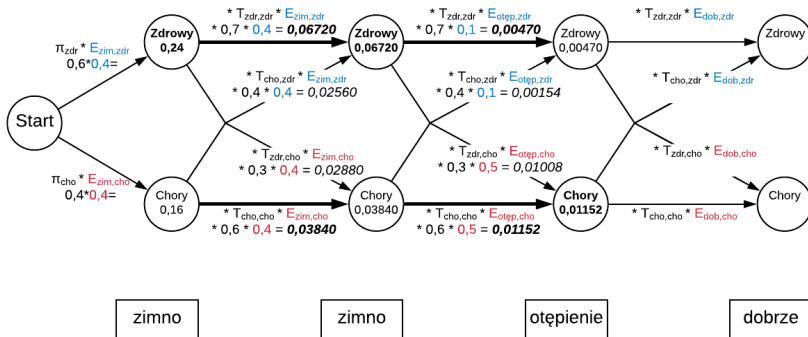
Wykres kratowy – *trellis*

Viterbi – example



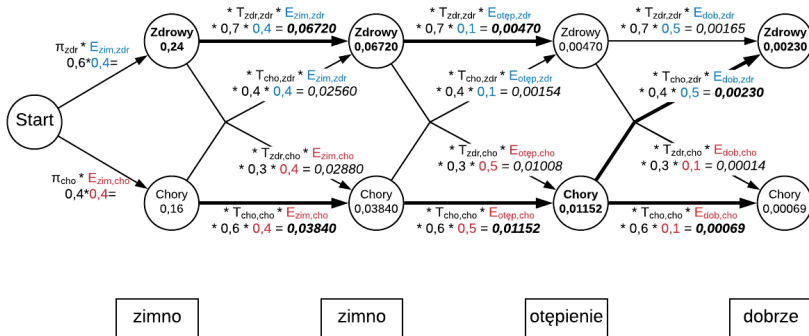
Wykres kratowy – *trellis*

Viterbi – example



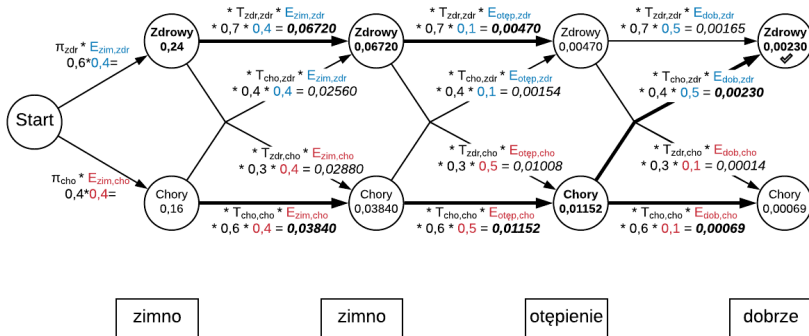
Wykres kratowy – *trellis*

Viterbi – example



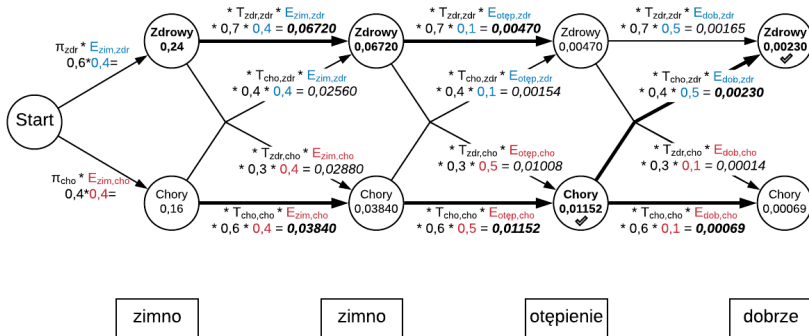
Wykres kratowy – *trellis*

Viterbi – example



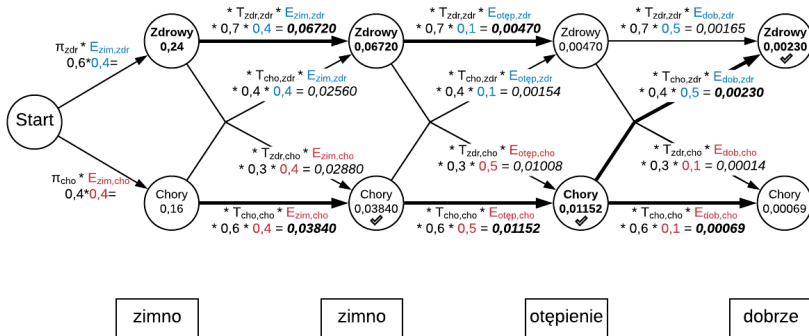
Wykres kratowy – *trellis*

Viterbi – example



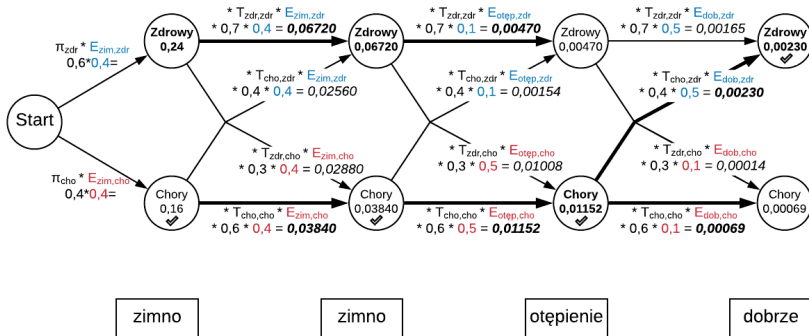
Wykres kratowy – *trellis*

Viterbi – example



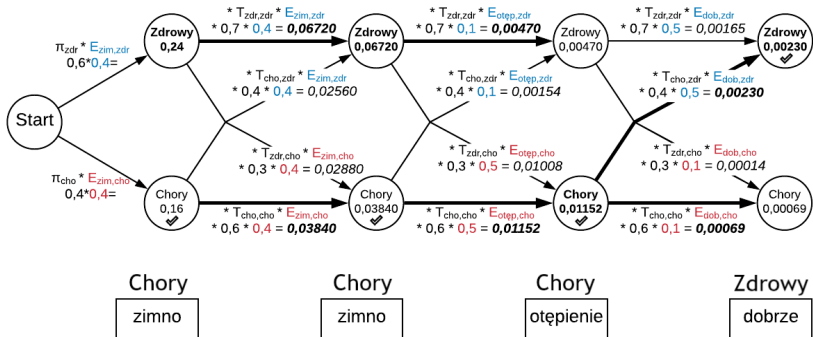
Wykres kratowy – *trellis*

Viterbi – example



Wykres kratowy – *trellis*

Viterbi – solution

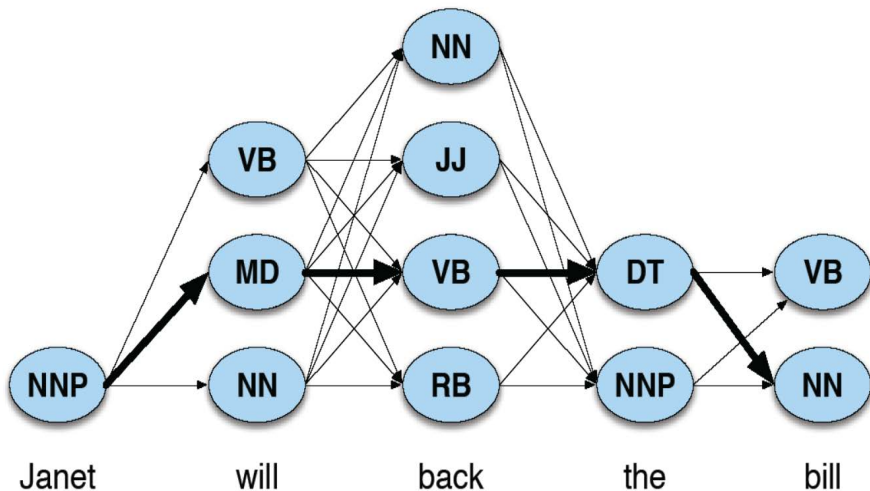


Wykres kratowy – *trellis*

Ukryty model Markowa dla POS-tagging

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

POS-tagging as an HMM



- Cel: rozumienie i/lub generowanie tekstów w języku naturalnym
- Wykorzystują techniki uczenia głębokiego (DL)
- Typy:
 - encoder
 - decoder
 - encoder-decoder

Modele typu *encoder* (Encoder-Only Models) np. BERT*

- Architektura typu encoder skupia się na przekształceniu tekstu wejściowego w zwięzłą reprezentację wektorową, zwaną *embeddings* (pol. *osadzenia*, *zanurzenia*). Uwzględnia obustronny kontekst
- Wyrazy o podobnym znaczeniu dostają reprezentacje bliskie w przestrzeni wektorowej
- Wykorzystywane w zadaniach wymagających rozumienia i analizy tekstu: klasyfikacja, analiza sentymentu (*badanie wydźwięku emocjonalnego wypowiedzi*), NER (Named-entity recognition)

*) Bidirectional Encoder Representations from Transformers – opracowany w Google AI Language w 2018 r.

W październiku 2020 r. zaadaptowany w Google Search dla ponad 70 języków

Modele typu *decoder* (Decoder-Only Models) np. GPT-4*

- Architektura typu decoder pozwala na generowanie tekstu przewidując kolejny wyraz sekwencji wyrazów. Uwzględnia jedynie lewostronny kontekst
- Dla zadanego wejścia (*promptu*) generowana jest wypowiedź wyraz po wyrazie – każdorazowo wykorzystując dotychczasową sekwencję jako kontekst
- Wykorzystywane we wszelkich zadaniach generowania tekstu, autouzupełnianiu, chatbotach

*) Generative Pre-trained Transformer – pierwszy model GPT-1 przedstawiony przez OpenAI w 2018 roku

Modele typu *encoder-decoder* np. T5* lub BART

- Architektura wykorzystująca encoder do przekształcenia tekstu na wejściu oraz decoder do generowania tekstu na wyjściu
- Tekst wejściowy przetwarzany jest na wektor kontekstu, który jest następnie wykorzystany do wygenerowania tekstu wyjściowego
- Stosowane w automatycznym streszczaniu tekstów, odpowiadaniu na pytania (*QA systems*), tłumaczeniu maszynowym

*) Text-to-Text Transfer Transformer – zbiór modeli LLM opracowanych w Google AI i przedstawionych w 2019 r. Niewielkie modyfikacje oryginalnej propozycji *modelu Transformer* opublikowanej przez Google w 2017 r.

Attention type and window size

Model	Rodzaj uwagi	Zasięg (rozmiar okna uwagi)
BERT-base/large	Pełna (global self-attention)	512 tokenów (każdy do każdego)
DistilBERT	Pełna	512 tokenów (jw.)
RoBERTa	Pełna	512 tokenów (jw.)
Longformer	Hybrydowa (local + global)	np. 512 lokalnie + globalnie nieliczne tokeny
BigBird	Sparse (random + local + global)	Mieszana: np. 512 lokalnie + próbki globalnie

Attention type and window size

- BERT wykorzystuje standardowe maskowane *self-attention*, które ma złożoność obliczeniową $O(n^2)$ względem długości sekwencji – dlatego standard to 512 tokenów
- Modele takie jak Longformer i BigBird modyfikują mechanizm *attention* (odpowiednio dopasowują rozmiar okna), dzięki czemu obsługują dłuższe sekwencje bez drastycznego wzrostu kosztów obliczeniowych – deklarowana złożoność liniowa względem długości sekwencji
- Longformer używa deterministycznych wzorców *attention* (okna + wybrane tokeny globalne)
- BigBird dodaje do tego możliwość zadania pewnej liczby losowych połączeń *attention*

Attention type and window size

Model	Rodzaj uwagi	Zasięg* (rozmiar okna uwagi)
GPT-2	Pełna, maskowana	1 024
GPT-3	Pełna, maskowana	2 048
GPT-3.5	Pełna, maskowana	do 16 385
GPT-4	Pełna, maskowana	do 32 768
GPT-4o	Pełna, maskowana (efektywnie zoptymalizowana)	do 128 000

***) każdy token widzi każdy z lewej – lewostronny kontekst**

I. Dana jest następująca kolekcja ucząca zdań:

$\langle s \rangle ja \ i \ ja \langle /s \rangle$ $\langle s \rangle ty \ to \ ja \langle /s \rangle$ $\langle s \rangle ja \ to \ ty \langle /s \rangle$ $\langle s \rangle ja \ to \ ty \ i \ to \langle /s \rangle$

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 $\langle s \rangle ty \ i \ to \ ja \langle /s \rangle$

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

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

sekw=(a, a, b); $\pi = 0,5 \ 0,5$

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