

Wstęp do NLP dla OAI

Witold Drzewakowski
15.01.2025

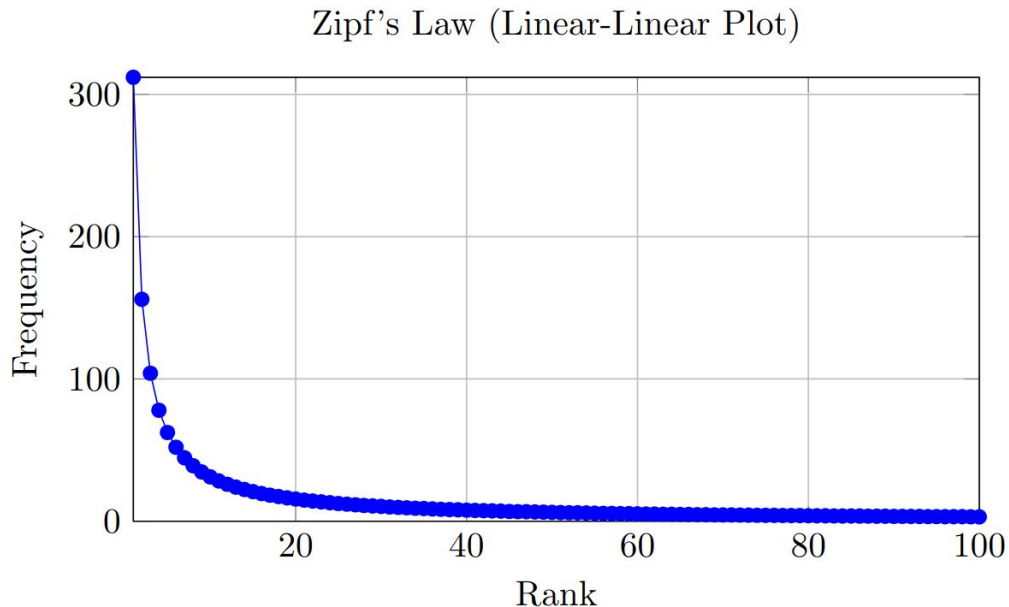
Cechy języka naturalnego (1)

- Prawo Zipfa

Komentarz:

Slajd wprowadza empiryczne prawo dotyczące częstości występowania słów w tekstach. W bardzo dużym uproszczeniu: występuje bardzo mało słów o dużej, ale szybko malejącej częstości i bardzo dużo słów o częstości występowania bardzo małej.

Prawo Zipfa wskazuje pewne konkretne przybliżenie tych częstości.



Cechy języka naturalnego (2)

- Różnicowanie

Słabość tego rodzaju można by nazwać niewinną, bo jest niebezpieczna z jednego tylko względu: grozi tym, że człowiek może się stać nudziarzem. Świat bowiem nie interesuje się na ogół przyczyną jakiegoś jawnego czynu, tylko jego skutkami. Wbrew mylnym pozorom ludzie nie wnikają w cudze pobudki.

– Joseph Conrad, Tajny agent (tłum. Aniela Zagórska)

*Na czerń postawie dziś
Nie spojrzysz tu i tak
A mnie nie rusza to
Bo płynę z wiatrem*

– Sylwia Grzeszczak, Motyle

Podamy teraz dwa bardzo ważne twierdzenia, które w połączeniu z Twierdzeniem 5.19 umożliwiają obliczanie wielu całek. Pierwsze z nich, twierdzenie o zamianie zmiennych, jest naturalnym uogólnieniem Twierdzenia 4.35 (o mierze liniowego obrazu zbioru mierzalnego) na przypadek odwzorowań nieliniowych.

– Paweł Strzelecki, Analiza matematyczna II (skrypt wykładu)

O czym ty do mnie rozmawiasz? Głowa cię nie boli?

– Chłopaki nie płaczą, reż. Olaf Lubaszenko

Cechy języka naturalnego (3)

Komentarz:

Zasada kompozycyjności głosi, że znaczenie wyrażenia jest zawarte w znaczeniu jego poszczególnych części oraz tego jak są one ze sobą połączone.

Dodatkową konsekwencją tej zasady jest możliwość konstruowania dowolnie długich, sensownych zdań!

- Kompozycyjność

Kasia czyta książkę.

Kasia czyta ciekawą książkę o podróżach.

Kasia czyta ciekawą książkę o podróżach w parku.

Kasia czyta ciekawą książkę o podróżach w parku, ponieważ planuje wycieczkę.

Zadania w NLP

Komentarz:

Od komputerów rozumiejących język naturalny oczekujemy, że będą nam pomagały w bardzo wielu dość różnych zadaniach. Na tym slajdzie wymieniono kilka z nich.

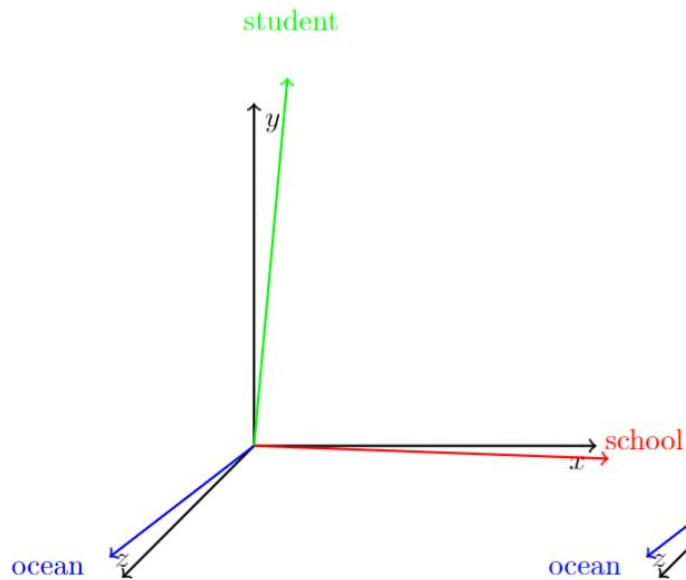
Task	Input	Output
Sentiment Analysis	I love studying AI!	Positive
POS Tagging	I love studying AI!	PRON VERB VERB NOUN PUNCT
Parsing	I love studying AI!	<pre>graph TD S --> NP1[NP] S --> VP1[VP] NP1 --> PRON1[PRON] PRON1 --> I1[I] VP1 --> V1[V] V1 --> love1[love] VP1 --> NP2[NP] NP2 --> V2[V] V2 --> studying[studying] NP2 --> N[N] N --> AI[AI] studying --- AI</pre>
Machine Translation	I love studying AI!	Ich liebe das Studieren von KI!
Chatbot	Hi, how are you?	I'm fine, thank you!
Named Entity Recognition	Apple is a company.	(Apple, ORGANIZATION)

Wektory słów

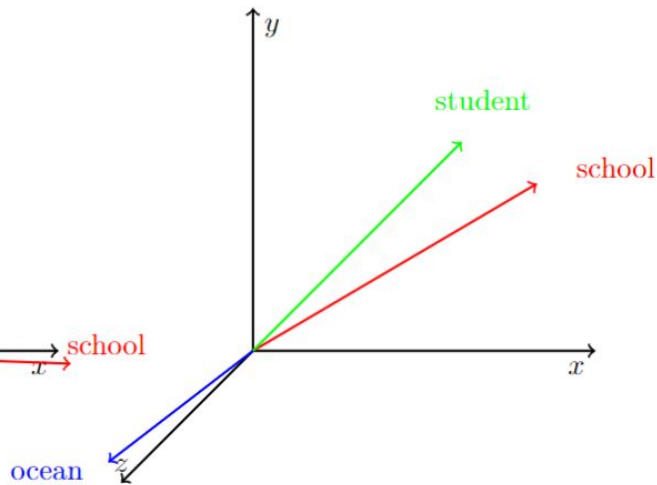
Komentarz:

Po lewej stronie zaprezentowano przykładowe wektory "rzadkie". Są one takiego wymiaru jak liczba słów w słowniku, pełne zer poza jednym miejscem gdzie znajduje się 1.

Takie wektory, choć pozwalają na odróżnienie od siebie słów, to nie pozwalają na dużo więcej ciekawych rzeczy. Bardziej ciekawe są wektory "gęste", u których bliskość w pewnym sensie między dwoma wektorami oddaje pewnego rodzaju podobieństwo wyjściowych słów.



Word	x	y	z
School	1.0	0.0	0.0
Student	0.0	1.0	0.0
Ocean	0.0	0.0	1.0



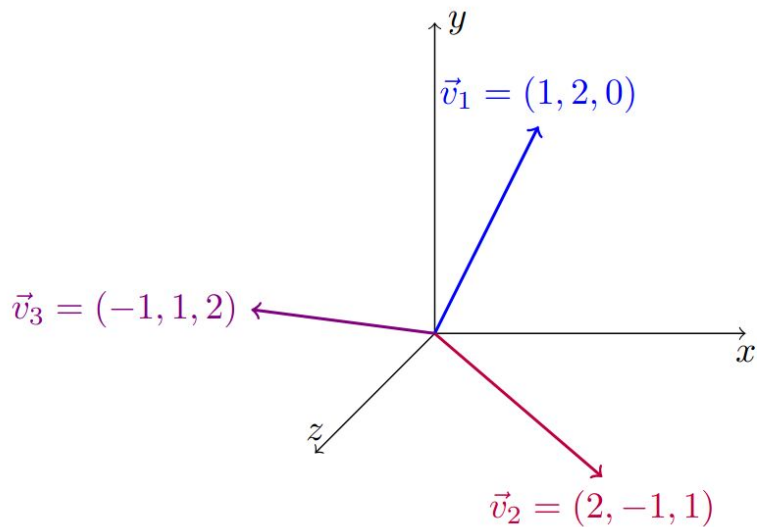
Word	x	y	z
School	0.34	0.21	0.01
Student	0.25	0.25	0.02
Ocean	0.07	0.05	0.45

Przypomnienie: mnożenie macierzy

Przypomnienie: mnożenie macierzy (1)

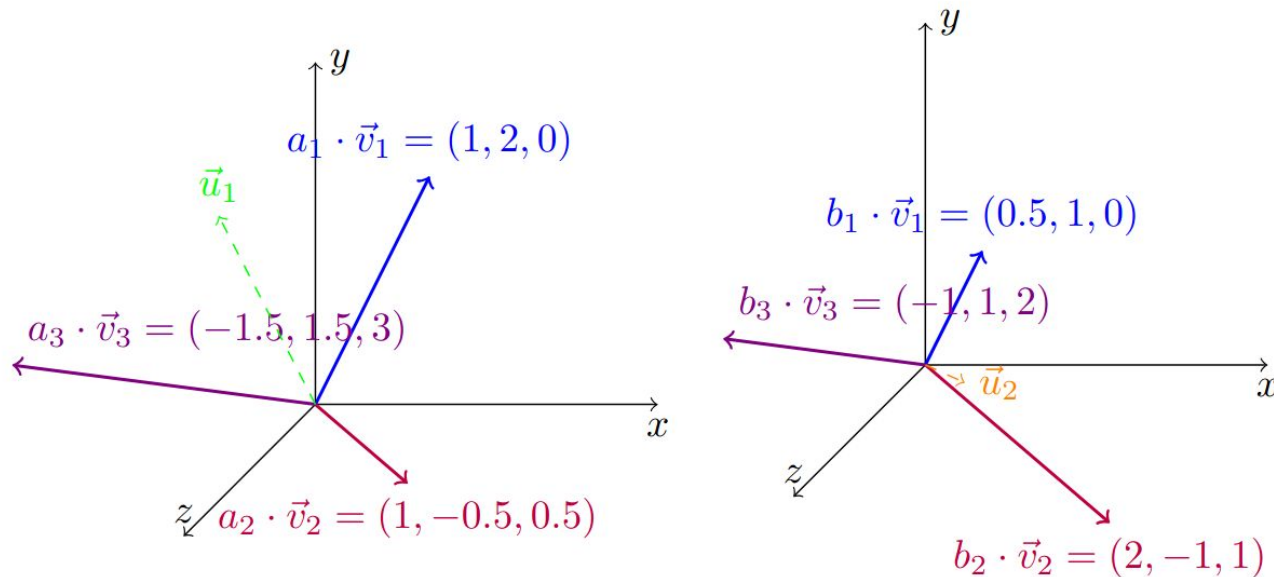
$$\begin{bmatrix} 1 & 2 & -1 \\ 2 & -1 & 1 \\ 0 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \\ 1.5 & 1 \end{bmatrix} = ?$$

Niech kolumny pierwszej macierzy to $\vec{v}_1, \vec{v}_2, \vec{v}_3$, a kolumny drugiej to \vec{a}, \vec{b} .



Przypomnienie: mnożenie macierzy (2)

Używamy współrzędnych wektora \vec{a} do przeskalowania wektorów $\vec{v}_1, \vec{v}_2, \vec{v}_3$, a następnie sumujemy otrzymując \vec{u}_1 . Analogicznie działamy wektorem \vec{b} otrzymując \vec{u}_2 .



Przypomnienie: mnożenie macierzy (3)

$$\begin{bmatrix} 1 & 2 & -1 \\ 2 & -1 & 1 \\ 0 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \\ 1.5 & 1 \end{bmatrix} = ?$$

Niech kolumny pierwszej macierzy to $\vec{v}_1, \vec{v}_2, \vec{v}_3$, a kolumny drugiej to \vec{a}, \vec{b} .

$$\vec{u}_1 = a_1 \cdot \vec{v}_1 + a_2 \cdot \vec{v}_2 + a_3 \cdot \vec{v}_3 = (0.5, 3, 3.5)$$

$$\vec{u}_2 = b_1 \cdot \vec{v}_1 + b_2 \cdot \vec{v}_2 + b_3 \cdot \vec{v}_3 = (1.5, 1, 3)$$

Wynik to:

$$\begin{bmatrix} \vec{v}_1 & \vec{v}_2 & \vec{v}_3 \end{bmatrix} \cdot \begin{bmatrix} \vec{a} & \vec{b} \end{bmatrix} = \begin{bmatrix} \vec{u}_1 & \vec{u}_2 \end{bmatrix} = \begin{bmatrix} 0.5 & 1.5 \\ 3 & 1 \\ 3.5 & 3 \end{bmatrix}$$

Przypomnienie: mnożenie macierzy (4)

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} = \begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix}$$

Przypomnienie: mnożenie macierzy (4)

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} = \begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{a}_{11} & \mathbf{a}_{12} & \mathbf{a}_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{b}_{11} & b_{12} \\ \mathbf{b}_{21} & b_{22} \\ \mathbf{b}_{31} & b_{32} \end{bmatrix} = \begin{bmatrix} \mathbf{c}_{11} & ? \\ ? & ? \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{a}_{11} & \mathbf{a}_{12} & \mathbf{a}_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & \mathbf{b}_{12} \\ b_{21} & \mathbf{b}_{22} \\ b_{31} & \mathbf{b}_{32} \end{bmatrix} = \begin{bmatrix} c_{11} & \mathbf{c}_{12} \\ ? & ? \end{bmatrix}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ \mathbf{a}_{21} & \mathbf{a}_{22} & \mathbf{a}_{23} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{b}_{11} & b_{12} \\ \mathbf{b}_{21} & b_{22} \\ \mathbf{b}_{31} & b_{32} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ \mathbf{c}_{21} & ? \end{bmatrix}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ \mathbf{a}_{21} & \mathbf{a}_{22} & \mathbf{a}_{23} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & \mathbf{b}_{12} \\ b_{21} & \mathbf{b}_{22} \\ b_{31} & \mathbf{b}_{32} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & \mathbf{c}_{22} \end{bmatrix}$$

“Znaczenie słowa to jego użycie w języku.”

Die Bedeutung eines Wortes ist sein Gebrauch in der Sprache.

— Ludwig Wittgenstein, *Dociekania filozoficzne* (*Philosophische Untersuchungen*), §43.

Word2vec (BoW)

Słabość tego rodzaju można by nazwać niewinną, ...

$$\begin{array}{rcl} \text{tego} & \rightarrow & (0, 0, \dots, 0, 1, 0, \dots, 0) \\ \text{rodzaju} & \rightarrow & (0, 0, \dots, 0, 0, \dots, 1, 0) \\ \text{by} & \rightarrow & (0, 1, \dots, 0, 0, 0, \dots, 0) \\ \text{nazwać} & \rightarrow & (0, 0, \dots, 1, 0, 0, \dots, 0) \\ \hline + & & (0, 1, \dots, 1, 1, 0, \dots, 0) \end{array}$$

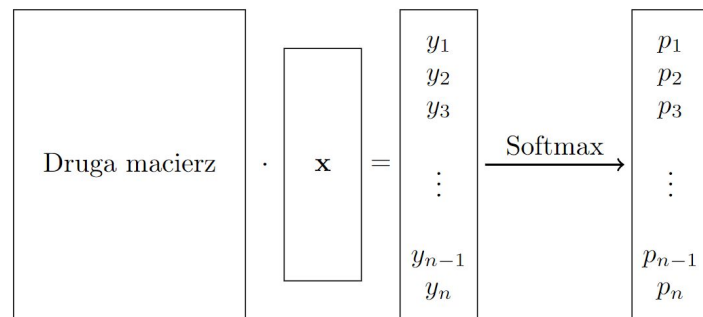
$$\begin{array}{|c|} \hline \text{Macierz embeddingów} \\ \hline \end{array} \cdot \begin{array}{c} 0 \\ 1 \\ \vdots \\ 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{array} = \begin{array}{|c|} \hline \mathbf{x} \\ \hline \end{array}$$

Komentarz:

Po kolei przechodzimy przez cały korpus danych tekstowych, ustalając center word i okno zawierające słowa blisko niego. W algorytmie mamy dwie wyuczalne macierze (ich komórki zmieniają się w czasie treningu).

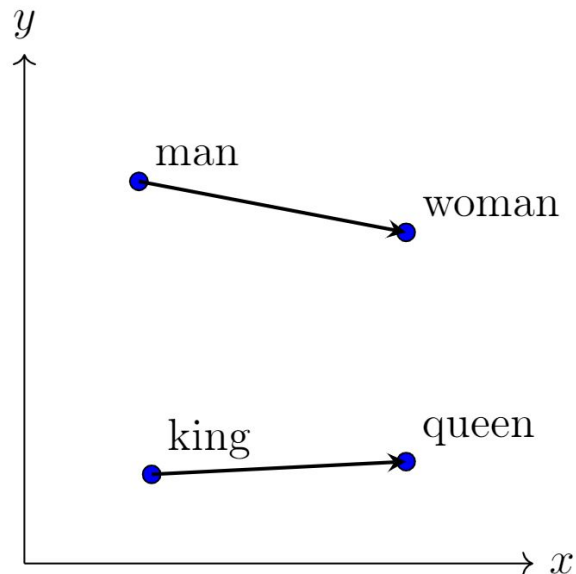
Wektor liczb rzeczywistych (y_1, y_2, \dots, y_n) ma liczbę wymiarów równą liczbie unikalnych słów w modelu. Zamieniamy go na wektor liczb dodatnich sumujących się do 1, używając funkcji Softmax. Takie wektory możemy traktować jak rozkłady prawdopodobieństwa.

W czasie trenowania modelu Word2Vec, zmieniane są komórki obu macierzy tak, aby zwiększyć prawdopodobieństwo otrzymania słowa centralnego (w tym przykładzie "można").



$p_{\text{jabłko}} \searrow$ $p_{\text{telefon}} \searrow$ \dots $p_{\text{duży}} \searrow$ $p_{\text{można}} \nearrow$ $p_{\text{auto}} \searrow$ \dots $p_{\text{dom}} \searrow$

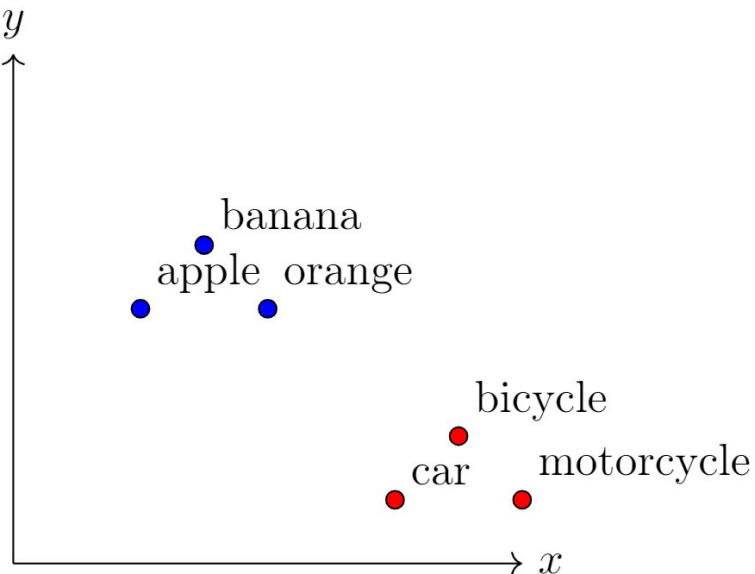
Cechy wektorów słów



Komentarz:

Po wyuczeniu wektorów słów algorytmem Word2Vec zaobserwowano, że:

- a) W przestrzeni znajdują się kierunki odpowiadające pewnym koncepcjom: np. $\text{queen} - \text{king} + \text{man} \approx \text{woman}$
- b) Słowa występujące w podobnych kontekstach znajdują się "blisko" siebie w przestrzeni wektorów.



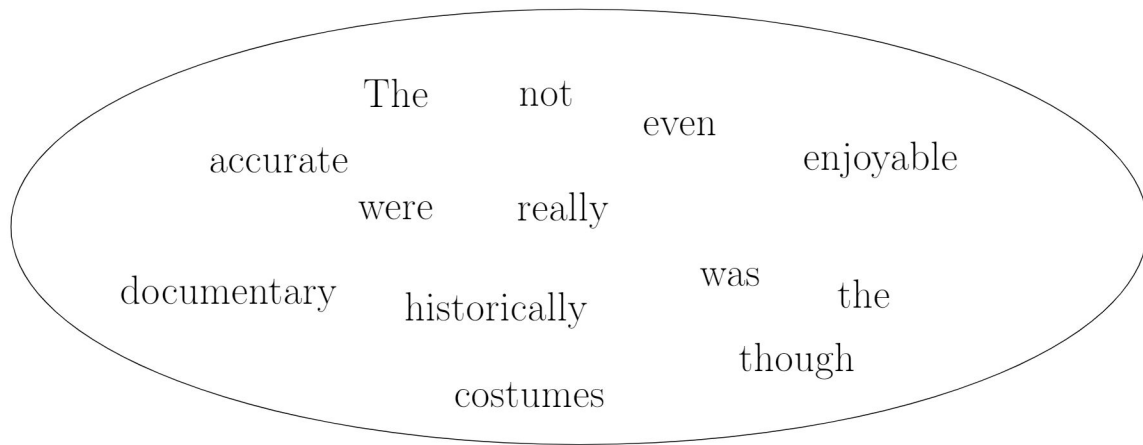
Bag-of-words

Komentarz:

Podano przykłady dwóch zdań, które mają dokładnie te same słowa, ale przez ich inną kolejność, znaczą co innego. W tym przypadku są to fragmenty recenzji filmów dokumentalnych o odmiennym sentymencie.

Ilustruje to fakt, że same wektory słów, które przypisują do danego słowa zawsze ten sam wektor niezależnie od kontekstu, w którym dane słowo się pojawia. nie pozwalają czasem na zrozumienie w pełni sensu całych zdań.

- Ale znaczenie zdania zależy też od kolejności, w jakiej pojawiają się słowa!



The documentary was enjoyable, even though the costumes were not really historically accurate.

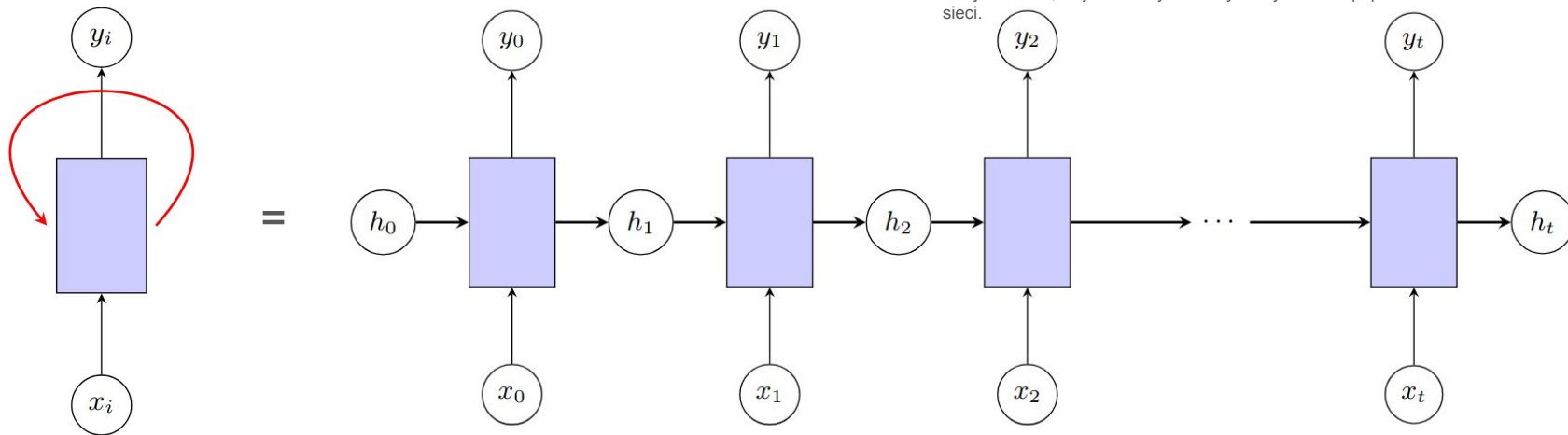
Even though the costumes were historically accurate, the documentary was not really enjoyable.

Rekurencyjne Sieci Neuronowe

Komentarz:

Sieci rekurencyjne to takie sieci, które przyjmują element wejścia (np. jedno słowo) oraz stan, a zwracają rezultat (np. klasyfikację danego słowa, następne słowo, albo nic) oraz następny stan.

Tą samą sieć uruchamia się tyle razy ile słów w zdaniu, podając jako wejście kolejne słowa, a jako stany – stany otrzymane z poprzednich uruchomień sieci.



$$h_i = \tanh(W_{hh}h_{i-1} + W_{hx}x_i + b_h)$$

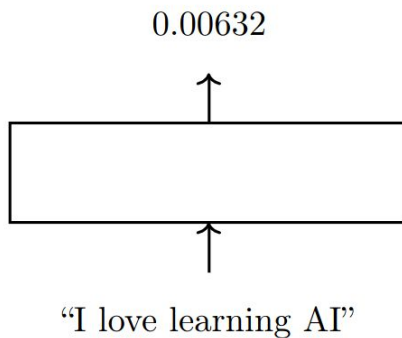
$$y_i = W_{yh}h_i + b_y$$

Modele języka

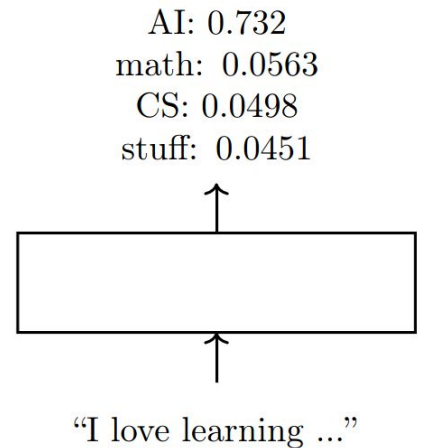
Czym są modele języka?

Modele języka pozwalają nam

- Przewidywać kolejne słowa
- Przypisywać prawdopodobieństwo do słów lub całych zdań



(a) $p((x_1, x_2, \dots, x_n))$



(b) $p(x_n | (x_1, x_2, \dots, x_{n-1}))$

N-gramy

Komentarz:

Modele oparte o N-gramy to najprostsze możliwe modele języka. Określają one prawdopodobieństwo następnego słowa na podstawie tego jak często po poprzedzających je N-1 słowach w danych treningowych występowało to właśnie słowo.

“How much more he told her as to his occupation
it was impossible for Winnie’s mother to discover.”

$$p_{\text{bigram}}(\text{“his”}|\text{“to”}) = 0.5, \quad p_{\text{bigram}}(\text{“discover”}|\text{“to”}) = 0.5,$$

$$p_{\text{trigram}}(\text{“as”}|\text{“told”, “her”}) = 1$$

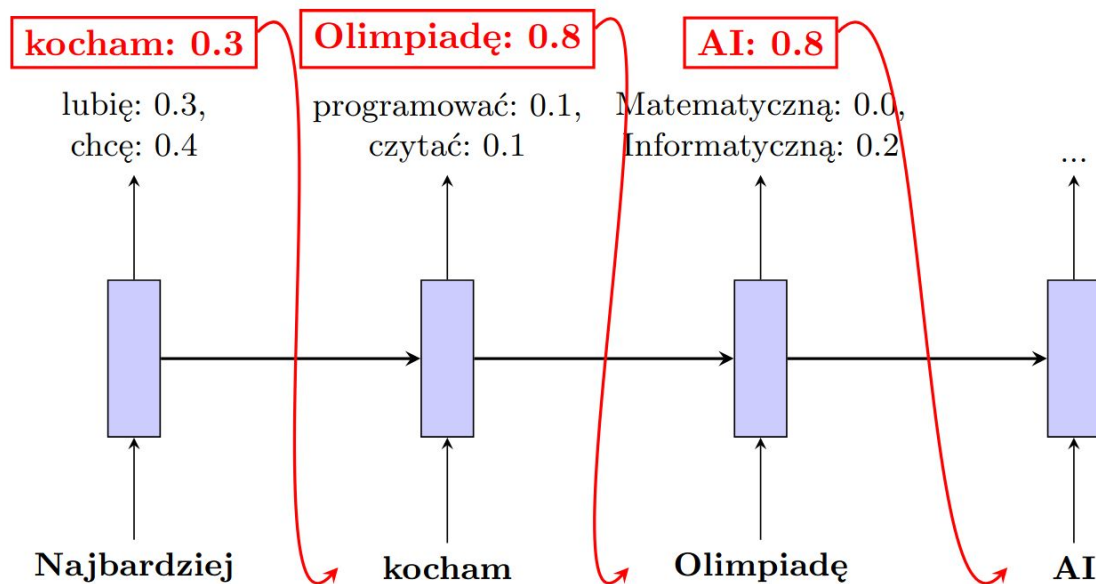
Bigrams with “to” as the first word: (to, his), (to, discover).

RNNy jako modele językowe

Komentarz:

Bardziej skomplikowane modele językowe mogą być wyuczone korzystając z RNNów. W tym wypadku wyjściem z sieci jest rozkład prawdopodobieństwa na następnym słowie.

Zauważmy, że poprzez przekazywanie stanu z poprzednich przejść przez sieć, model jest w stanie pamiętać kontekst dotychczasowej wypowiedzi.



Tokenizacja

```
example_text = "The Fulton County Grand Jury said Friday an investigation of Atlanta's recent primary election produced `` no evidence '' that any irregularities took place ."
```

```
print(word_tokenize(example_text))  
# OUT: ['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'Friday', 'an', 'investigation', 'of', 'Atlanta', "'s", 'recent', 'primary', 'election', 'produced', '', 'no', 'evidence', '', 'that', 'any', 'irregularities', 'took', 'place', '.']
```

```
print(nltk.wordpunct_tokenize(example_text))  
# OUT: ['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'Friday', 'an', 'investigation', 'of', 'Atlanta', "'", 's', 'recent', 'primary', 'election', 'produced', '', 'no', 'evidence', "'", 'that', 'any', 'irregularities', 'took', 'place', '.']
```

```
print(nltk.regexp_tokenize(example_text, pattern=r'\s+', gaps=True))  
# OUT: ['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'Friday', 'an', 'investigation', 'of', 'Atlanta's', 'recent', 'primary', 'election', 'produced', '', 'no', 'evidence', "'", 'that', 'any', 'irregularities', 'took', 'place', '.']
```

Komentarz:

Kiedy dzielimy zdanie na słowa, można to zrobić na wiele sposobów: trzeba podjąć decyzję np. czy znaki interpunkcyjne traktować jako oddzielne słowa.

W praktyce, współczesne modele językowe nie operują na słowach, tylko na ich krótszych fragmentach.

```
# tokenizer  
inputs = tokenizer("Tell me a nonobvious story.", return_tensors="pt")  
  
# decode inputs  
for i, token in enumerate(inputs['input_ids'][0]):  
    print(f"Token {i}: {token:<5} - {tokenizer.decode(token)}")
```

```
Token 0: 24446 - Tell  
Token 1: 502 - me  
Token 2: 257 - a  
Token 3: 1729 - non  
Token 4: 672 - ob  
Token 5: 1442 - vious  
Token 6: 1621 - story  
Token 7: 13 - .
```

Lista olimpijskich zadań z NLP

- (OAI-I-1) Zagadki [\[link\]](#)
- (OAI-I-1) Analiza zależnościowa [\[link\]](#)
- (OAI-I-2) Szyfry [\[link\]](#)
- (OAI-I-2) Tłumaczenie maszynowe [\[link\]](#)
- (IOAI-I-home) mBERT tuning [\[link\]](#)
- (IOAI-I-onsite) mBERT tuning, cont. [\[link\]](#)
- (IOAI-I-sample) Morphological inflection [\[link\]](#)
- (IOAI-I-sample) Biased embeddings [\[link\]](#)