Wstęp do NLP dla OAI

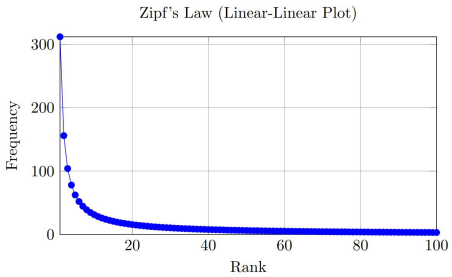
Witold Drzewakowski
13.12.2025

Roadmap

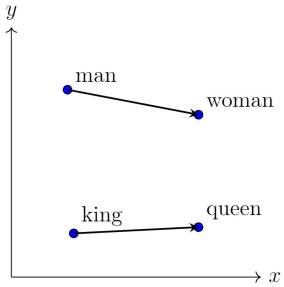
Komentarz:

Główne obszary NLP, których uczy Olimpiada na różnych etapach (1, 2, 3, obóz przygotowawczy). W tym wykładzie zostaną omówione podstawy NLP oraz Wektory słów.

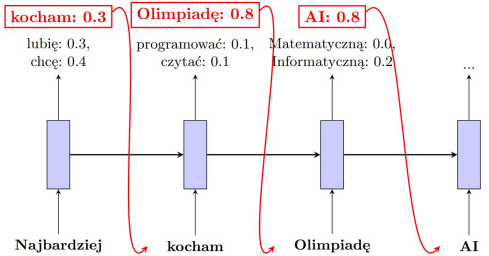
1. Podstawy NLP



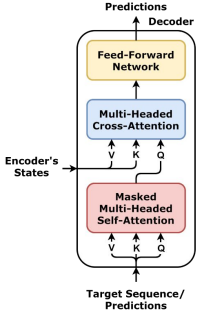
2. Wektory słów



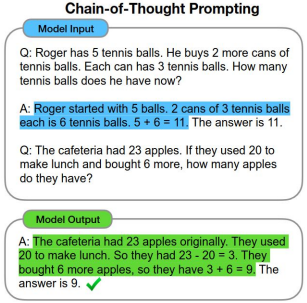
3. Rekurencyjne Sieci Neuronowe



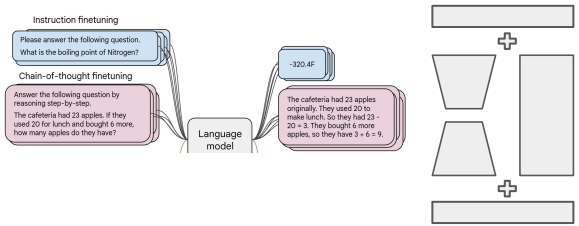
4. Atencja i transformery



5. Duże modele językowe (LLM)



6. Dostrajanie LLMów



Krótka historia chatbotów

```
  Welcome to
                EEEEEE  LL      IIII   ZZZZZZ   AAAAA
                EE      LL       II       ZZ   AA   AA
                EEEEE   LL       II     ZZZ    AAAAAAA
                EE      LL       II    ZZ      AA   AA
                EEEEEE  LLLLLL  IIII ZZZZZZ    AA   AA

  Eliza is a mock Rogerian psychotherapist.
  The original program was described by Joseph Weizenbaum in 1966.
  This implementation by Norbert Landsteiner 2005.


ELIZA: Is something troubling you ?
YOU:   Men are all alike.
ELIZA: What is the connection, do you suppose ?
YOU:   They're always bugging us about something or other.
ELIZA: Can you think of a specific example ?
YOU:   Well, my boyfriend made me come here.
ELIZA: Is it important to you that your boyfriend made you come here ?
YOU:   He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
YOU:   It's true. I am unhappy.
ELIZA: Can you explain what made you unhappy ?
YOU:
```

Komentarz:

W latach 60. opracowano pierwszy w historii chatbot ELIZA. Chatbot nie rozumiał znaczenia zdań w żadnym sensie, działał przez odszukiwanie najważniejszego słowa w ostatniej wypowiedzi użytkownika, a następnie obracał jego zdanie tak, aby zapytać o tą najważniejszą rzecz.

Współczesne chatboty nie opierają się na tak bardzo ograniczonym podejściu ustalonych reguł, tylko uczone są na podstawie dużych zbiorów danych i w czasie treningu wykształcają zdolność wykonywania dużo trudniejszych przekształceń i analizy tekstu, np. pisania kodu programów komputerowych.

can you help me prepare tikz plot in tex that demonstrates an idea of cosine similarity?

Sure! Here is a clean, self-contained **TikZ figure** that visually explains cosine similarity by showing two vectors, the angle between them, and a projection. This works directly in any modern LaTeX document using `tikz` and `calc`.

Minimal Working TikZ Example

tex — Skopiuj kod

```tex
\begin{tikzpicture}[>=stealth, line cap=round, line join=round, scale=2]

% Axes
\draw[->, thick] (-0.2,0) -- (2.2,0) node[right] {$x$};
\draw[->, thick] (0,-0.2) -- (0,2.2) node[above] {$y$};
```

Zróżnicowanie języka naturalnego

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

Trudności w przetwarzaniu języka

- *Jan podszedł do człowieka z zakupami.* – Kto miał zakupy?
- *I saw her duck.* – W innych językach może być jeszcze trudniej :)
- *Podaj mi to.* – co?
- *superhipernieodpowiedzialność*
- *Colorless green ideas sleep furiously.* – syntaktycznie wszystko poprawnie. Ale cóżże miałoby to znaczyć?

Kompozycyjność języka

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!

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.

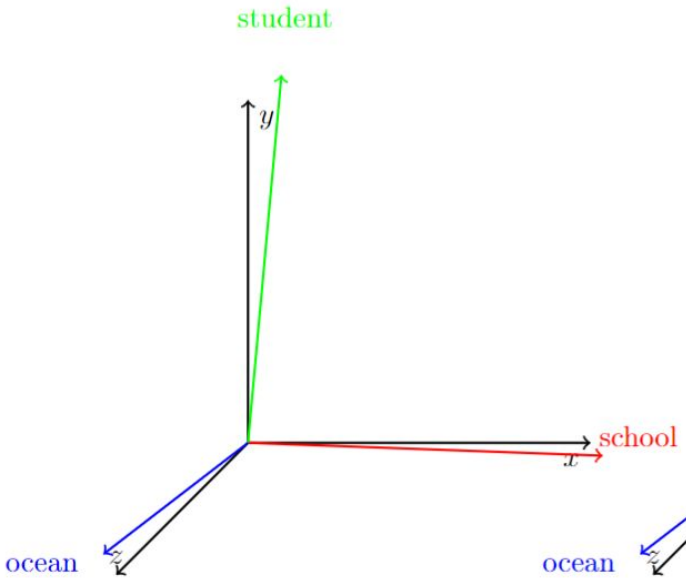
Nie stworzymy zaawansowanych i przydatnych systemów NLP w oparciu o ustalone zbiory reguł.

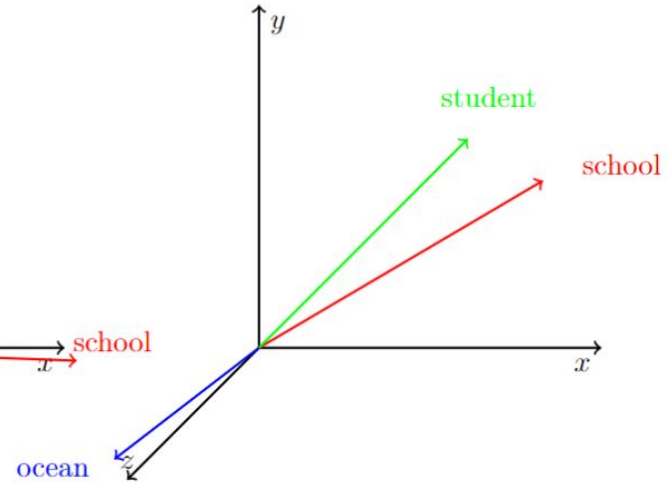
Wektory słów

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

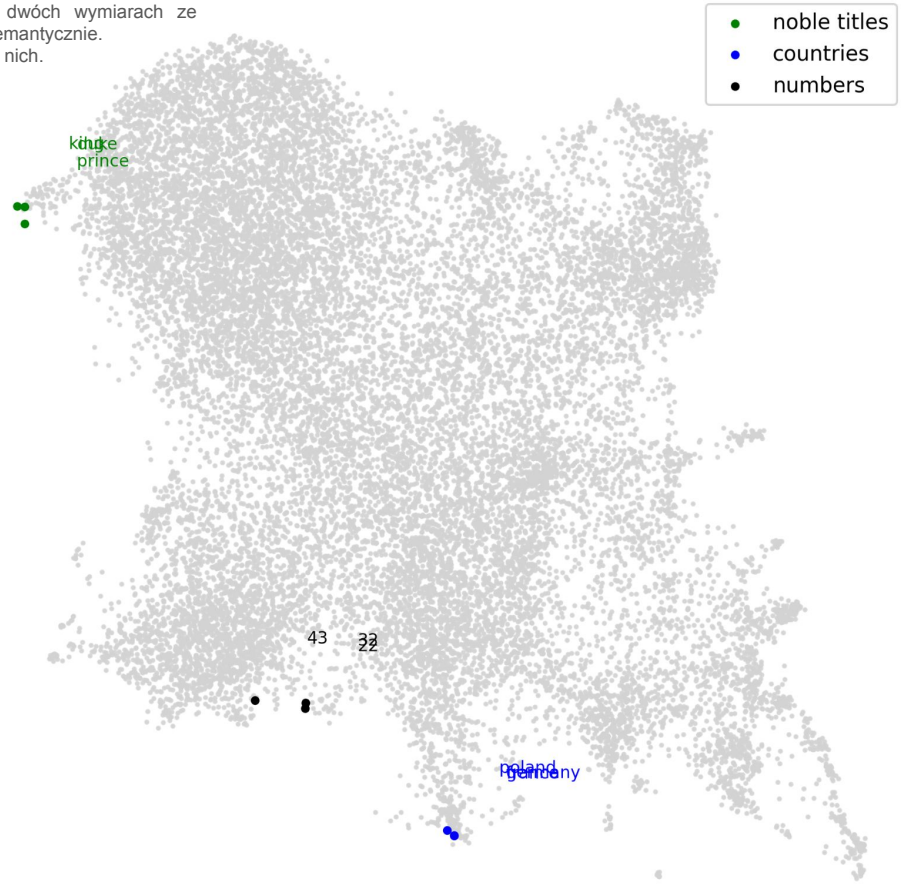
“Znaczenie słowa to jego użycie w języku.”

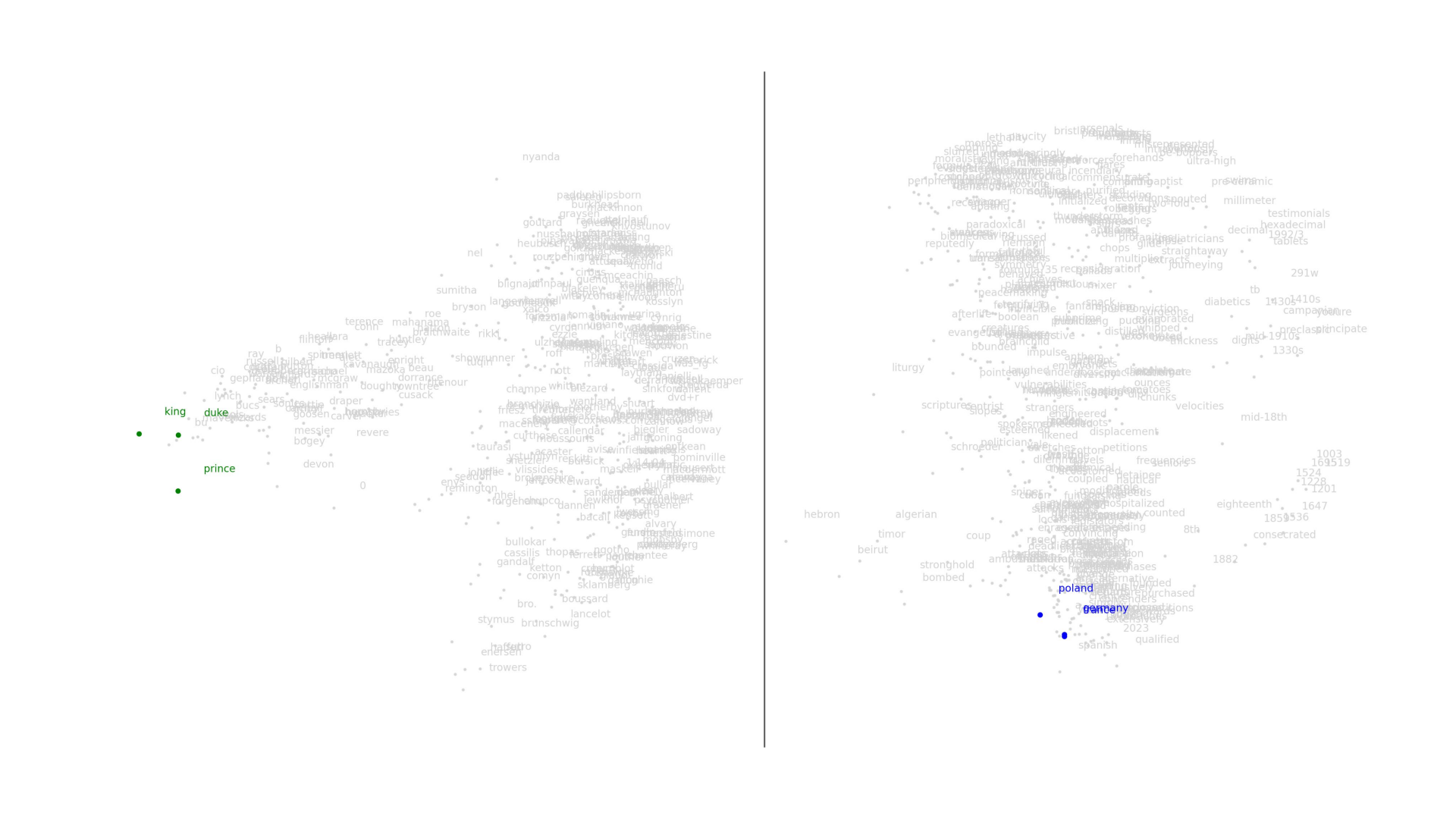
Die Bedeutung eines Wortes ist sein Gebrauch in der Sprache.

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

Komentarz:

Wizualizacja wytrenowanych wektorów GLOVE w dwóch wymiarach ze wskazanymi trzema klastrami o słowach podobnych semantycznie.
Na następnym slajdzie znajduje się zbliżenie na dwa z nich.



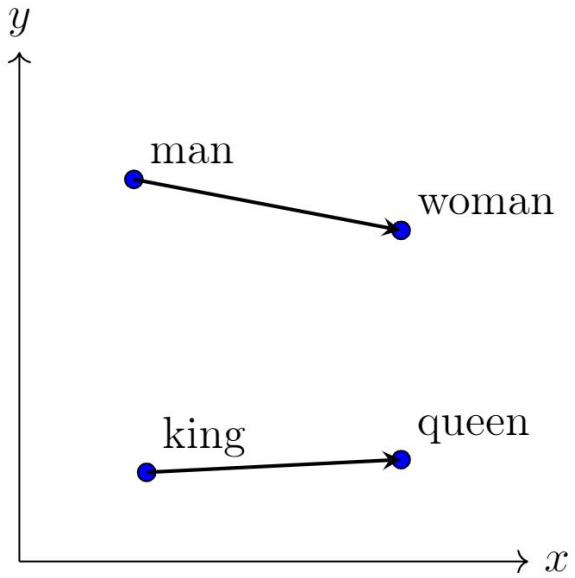

king
duke
prince
poland
germany

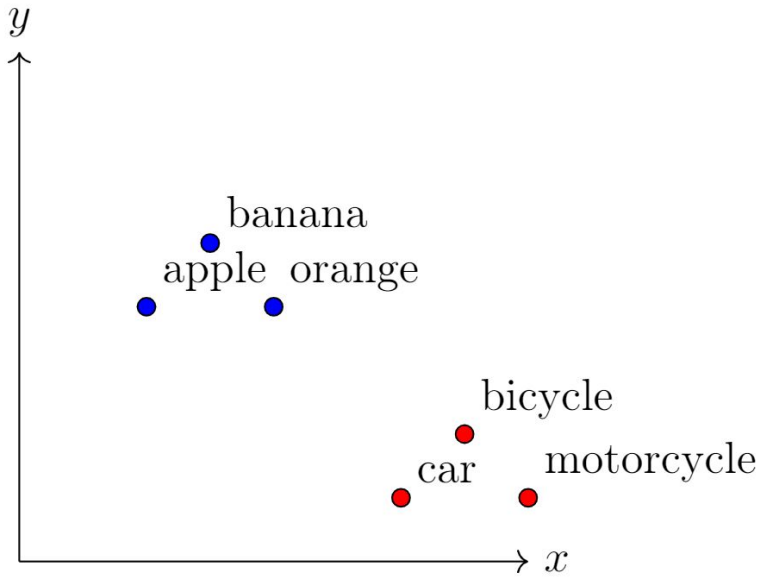
Cechy wektorów słów

Komentarz:

Po wyuczeniu wektorów słów algorytmem Word2Vec zaobserwowano, że:

a) W przestrzeni znajdują się kierunki odpowiadające pewnym koncepcjom: np. queen - king + man ~= woman
b) Słowa występujące w podobnych kontekstach znajdują się ”blisko” siebie w przestrzeni wektorów.



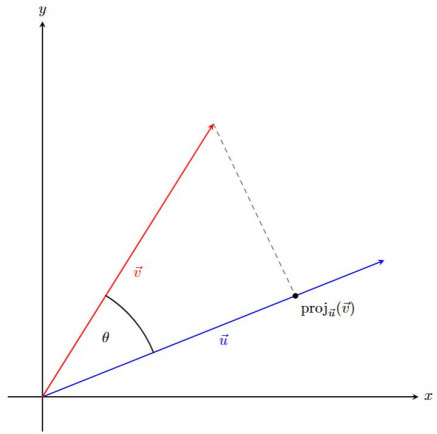


Przykład: Synonimy za pomocą wektorów słów

```python
def knn(vec, k=5):
    distances = [
        (other_word, cosine_similarity(vec, word_vectors[other_word]))
        for other_word in word_vectors.key_to_index.keys()
    ]
    distances.sort(key=lambda x: x[1], reverse=True)
    return distances[:k]


def print_synonyms(word):
    res = knn(word_vectors[word])
    print(f"Synonyms of '{word}':", [r for r, _ in res])


print_synonyms('school')  # equivalent to word_vectors.most_similar('school')
```



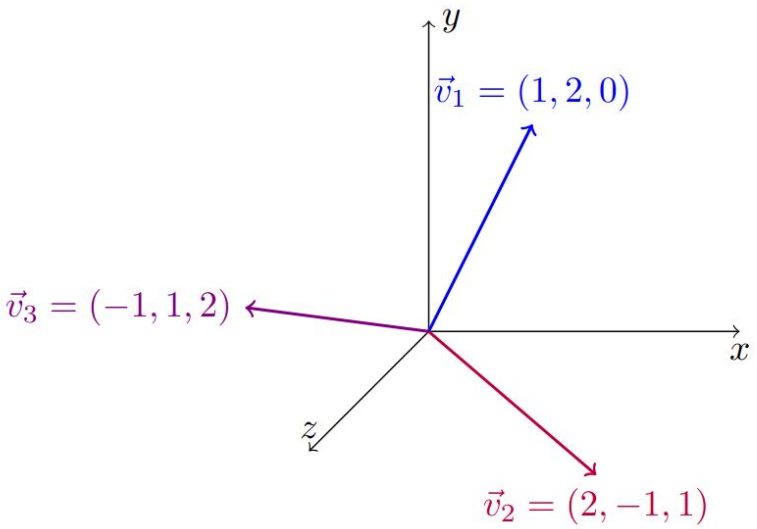
$$\cos(\theta) = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \, \|\vec{v}\|}$$

Przypomnienie: mnożenie macierzy

Przypomnienie: mnożenie macierzy (1)

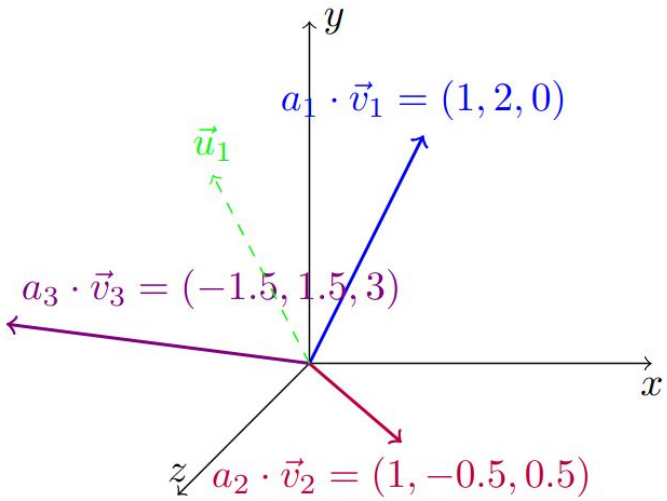
$$\begin{bmatrix} 1 & 2 & -1 \\ 2 & -1 & 1 \\ 0 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \\ 1.5 & 1 \end{bmatrix} = ?$$

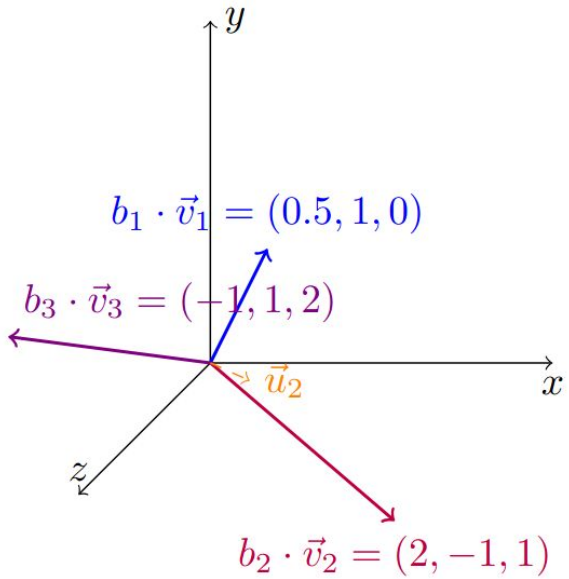
Niech kolumny pierwszej macierzy to $\vec{v}_1, \vec{v}_2, \vec{v}_3$, a kolumny drugiej to $\vec{a}, \vec{b}$.



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)

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

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

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

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

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

Word2Vec (BoW)

Słabość $\underbrace{\text{tego}}$ $\underbrace{\text{rodzaju}}$ $\boxed{\text{można}}$ $\underbrace{\text{by}}$ $\underbrace{\text{nazwać}}$ niewinną, ...

$$
\begin{aligned}
\boxed{\text{można}} &\rightarrow w \\
\underbrace{\text{tego}} &\rightarrow u_{\text{tego}} \\
\underbrace{\text{rodzaju}} &\rightarrow u_{\text{rodzaju}} \\
\underbrace{\text{by}} &\rightarrow u_{\text{by}} \\
\underbrace{\text{nazwać}} &\rightarrow u_{\text{nazwać}}
\end{aligned}
$$

$$u = u_{\text{tego}} + u_{\text{rodzaju}} + u_{\text{by}} + u_{\text{nazwać}}$$

$$\text{Softmax}(w^T u) \nearrow$$

Komentarz:

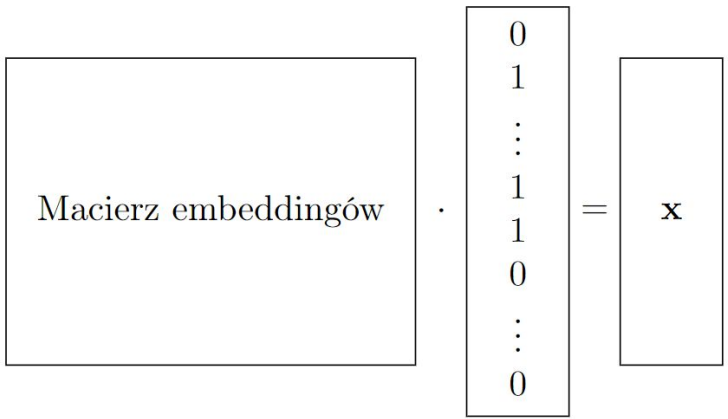
Po kolei przechodzimy przez cały korpus danych tekstowych, ustalając słowo centralne i okno zawierające słowa blisko niego. W algorytmie mamy dwie wyuczalne macierze (ich komórki zmieniają się w czasie treningu) - U i W.

W czasie trenowania modelu Word2Vec, zmieniane są komórki obu macierzy tak, aby zwiększyć bliskość słowa centralnego do sumy wektorów słów okalających (a oddalić od innych)

Word2Vec (BoW)

Słabość tego rodzaju **można** by nazwać niewinną, ...

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



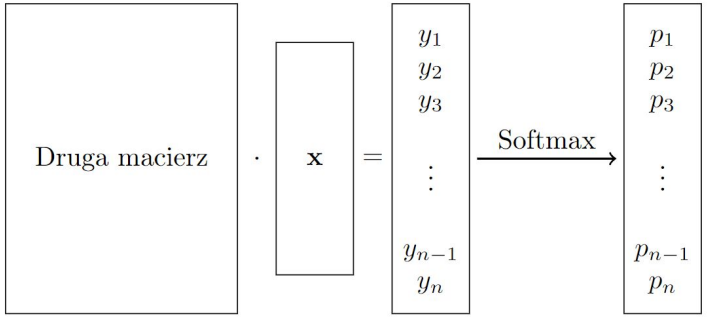
Komentarz:

Inna perspektywa na tą samą metodę.

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).

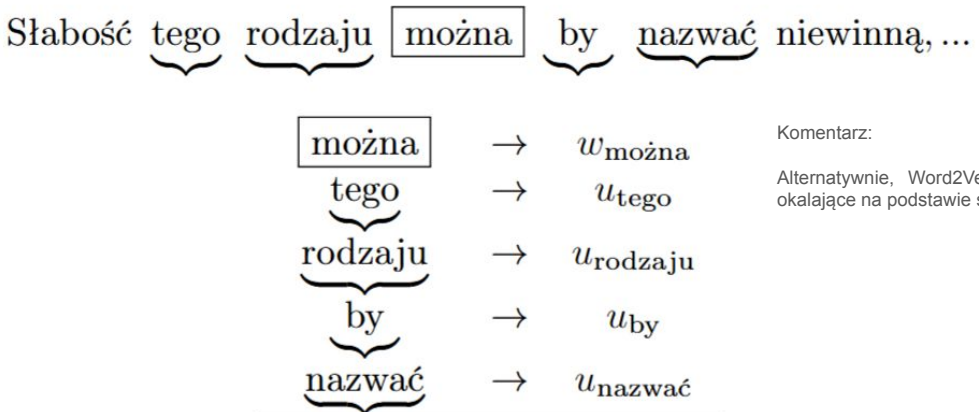
Wektor liczb rzeczywistych (y1, y2, …, yn) ma liczbę wymiarów równą liczbie unikalnych słów w modelu. Zamieniamy go na wektor liczb dodatnich sumujących się do 1, używając funkcji Softmax. Takie wektory możemy traktować jak rozkłady prawdopodobieństwa.

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



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

Word2Vec (skip gram)



Komentarz:

Alternatywnie, Word2Vec może być trenowany aby przewidywać słowa okalające na podstawie słowa centralnego.

$w_{\text{można}} u_{\text{tego}}$ ↗ $w_{\text{można}} u_{\text{rodzaju}}$ ↗ $w_{\text{można}} u_{\text{by}}$ ↗ $w_{\text{można}} u_{\text{nazwać}}$ ↗

Ale tak naprawdę zwiększamy wartość softmax, co wymaga policzenia wszystkich prawdopodobieństw!

Word2Vec (skip gram) – negative sampling

Słabość tego rodzaju **można** by nazwać niewinną, ...

można	$\rightarrow$	$w_{\text{można}}$
tego	$\rightarrow$	u_{tego}
rodzaju	$\rightarrow$	u_{rodzaju}
by	$\rightarrow$	u_{by}
nazwać	$\rightarrow$	$u_{\text{nazwać}}$
$\sim$ *kot*	$\rightarrow$	u_{kot}
$\sim$ *niebo*	$\rightarrow$	u_{niebo}
$\sim$ *dlaczego*	$\rightarrow$	u_{dlaczego}
$\sim$ *drzewo*	$\rightarrow$	u_{drzewo}

Komentarz:

Aby nie liczyć funkcji softmax, która wymaga policzenia podobieństw i prawdopodobieństw słowa centralnego do wszystkich słów w słowniku, można zastosować koncepcję negative samplingu:

losujemy 4 słowa które NIE są słowami okalającymi i zmniejszamy ich podobieństwo do słowa centralnego, a zwiększamy podobieństwo słów okalających. W ten sposób oszczędzamy moc obliczeniową.

$$w^T_{\text{można}} u_{\text{kot}} \searrow \quad w^T_{\text{można}} u_{\text{niebo}} \searrow \quad w^T_{\text{można}} u_{\text{dlaczego}} \searrow \quad w^T_{\text{można}} u_{\text{drzewo}} \searrow$$

$$w_{\text{można}} u_{\text{tego}} \nearrow \quad w_{\text{można}} u_{\text{rodzaju}} \nearrow \quad w_{\text{można}} u_{\text{by}} \nearrow \quad w_{\text{można}} u_{\text{nazwać}} \nearrow$$

Jakie są wady wektorów słów?

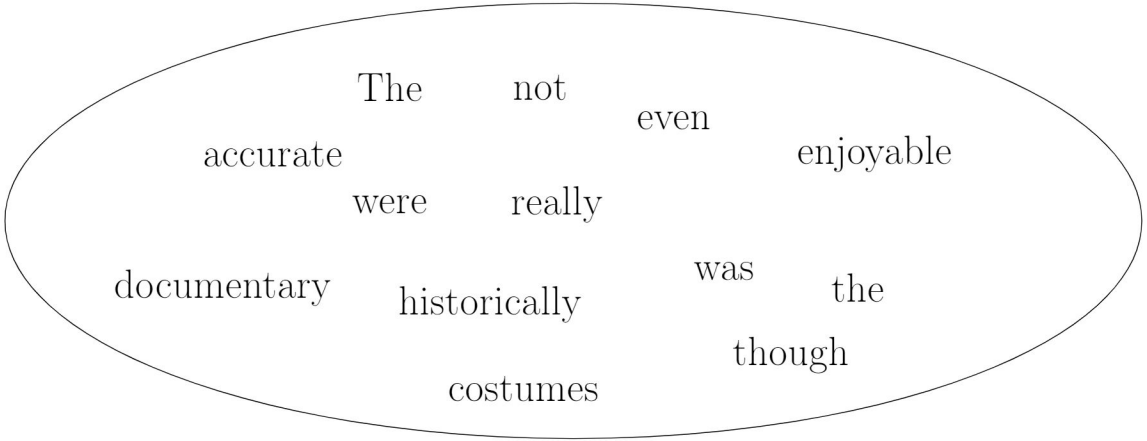
(1) Problem: Bag-of-words

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 dokumentarnych 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.

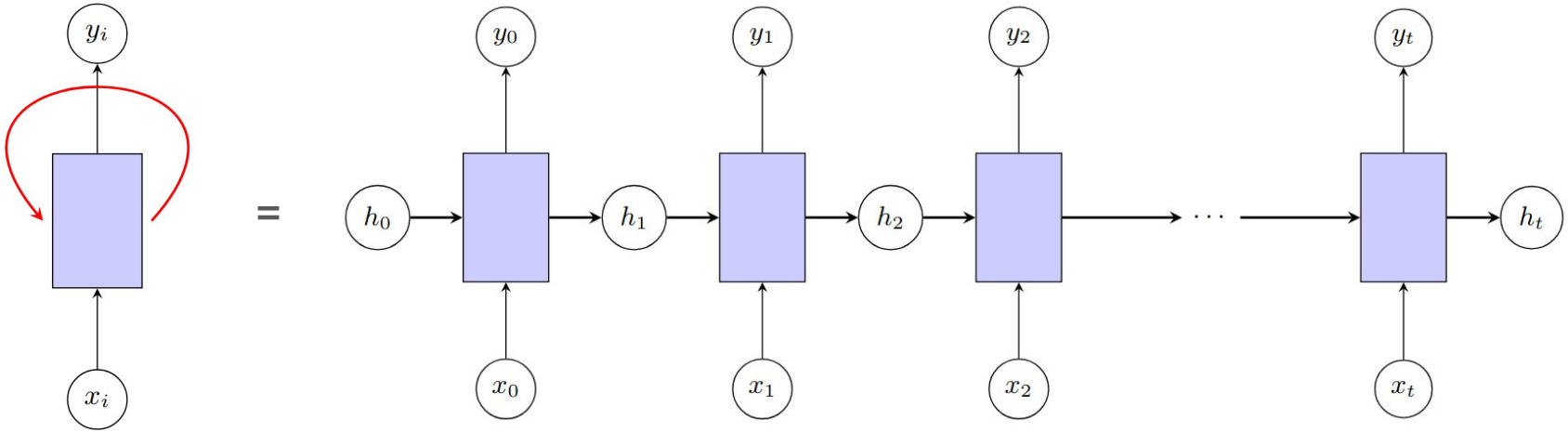
- Znaczenie zdania zależy też od kolejności, w jakiej pojawiają się słowa!



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.

(1) Rozwiązanie: Rekurencyjne Sieci Neuronowe



$$h_i = \tanh(W_{\text{hh}} h_{i-1} + W_{hx} x_i + b_{\text{h}})$$
$$y_i = W_{\text{yh}} h_i + b_{\text{y}}$$

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.

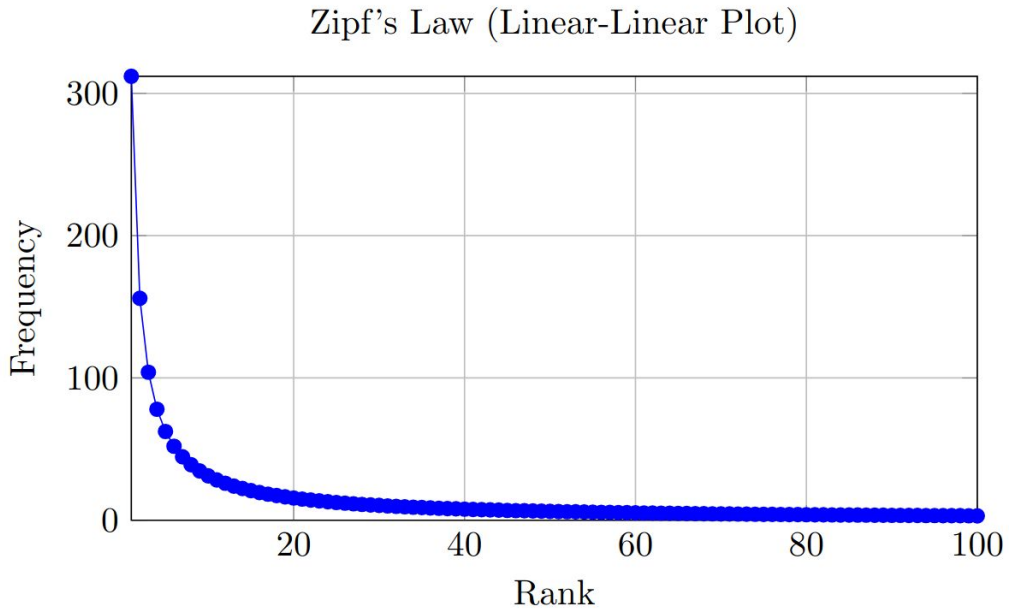
Tą samą sieć uruchamia się tyle razy ile słów w zdaniu, podając jako wejście kolejne słowa, a jako stany – stany otrzymane z poprzednich uruchomień sieci.

(2) Problem: Prawo Zipfa

Komentarz:

Slajd wprowadza empiryczne prawo dotyczące częstości występowania słów w tekstach. W bardzo dużym uproszczaniu: 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.



(2) Problem: Dostosowanie częstości słów

To counter the imbalance between the rare and frequent words, we used a simple subsampling approach: each word w_i in the training set is discarded with probability computed by the formula

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \tag{5}$$

Komentarz:

Word2Vec bierze poprawkę na to że niektóre słowa występują bardzo często, odrzucając niektóre słowa z prawdopodobieństwem zależnym od jego częstości.

(3) Problem: wielki słownik

1. Każde słowo to osobny element słownika.
2. Co jeśli jakieś słowo nie pojawiło się w korpusie treningowym?
    a. *superhipernieodpowiedzialność*
    b. Rindfleischetikettierungsüberwachungsaufgabenübertragungsgesetz

(3) Rozwiązanie: tokenizacja

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.

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

print(word_tokenize(example_text))
# OUT: ['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'Friday', 'an', 'inv
estigation', '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', 'inv
estigation', 'of', 'Atlanta', "'", 's', 'recent', 'primary', 'election', 'produc
ed', '``', 'no', 'evidence', "''", 'that', 'any', 'irregularities', 'took', 'pla
ce', '.']

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

```
# 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    - .
```

Modele języka

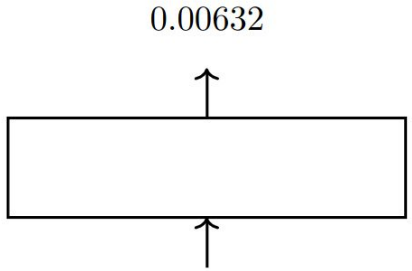
Czym są modele języka?

Komentarz:

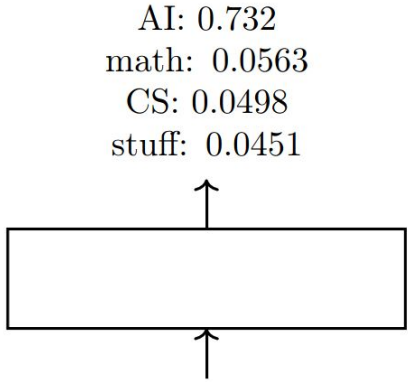
Modele języka są modelami probabilistycznymi, które dopasowują się do danych. W najprostszym podejściu, ich jakość mierzymy przez poziom dopasowania do danych testowych, np. przy użyciu metryki perplexity.

Modele języka pozwalają nam

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



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



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

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."

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

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

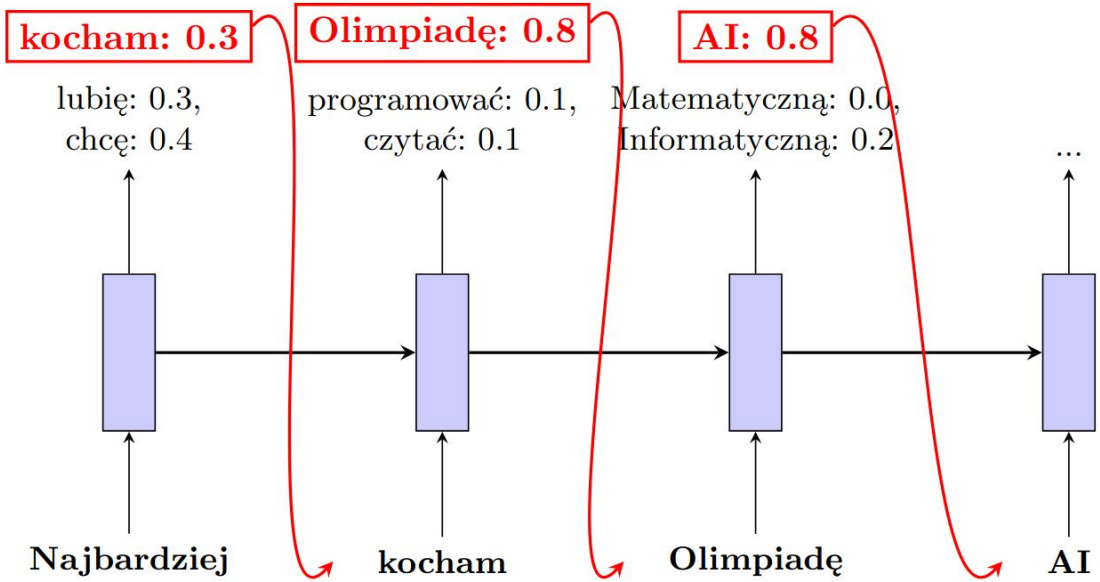
Bigrams with "to" as the first word: (to, his), (to, discover).

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.

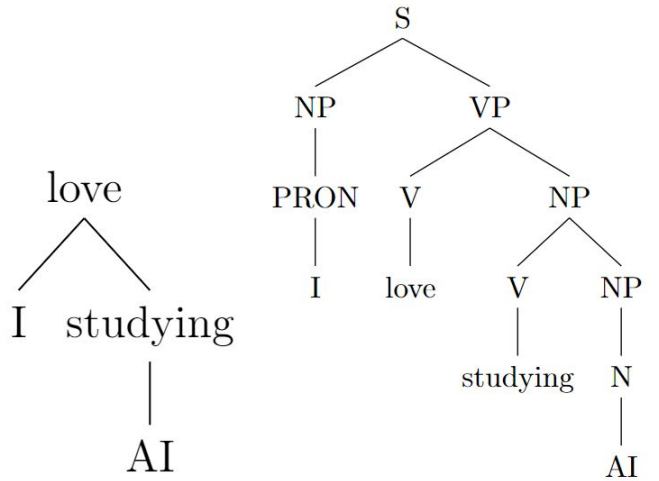
Zauważmy, że poprzez przekazywanie stanu z poprzednich przejść przez sieć, model jest w stanie pamiętać kontekst dotychczasowej wypowiedzi.



Standardowe 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!	
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)

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]
- (OAI-II-1) Wykrywanie halucynacji [link]
- (OAI-II-2) Ekstrakcja źródeł [link]
- (OAI-II-3) Stylizacja tłumaczeń
- (IOAI-I-home) mBERT tuning
- (IOAI-I-onsite) mBERT tuning, cont.
- (IOAI-I-sample) Morphological inflection
- (IOAI-I-sample) Biased embeddings