Wektorowe reprezentacje dystrybucyjne (*embeddings*)

Agnieszka Ławrynowicz

Wydział Informatyki Politechniki Poznańskiej

23 marca 2021

Wektorowa semantyka

Reprezentacja tekstu

- podstawowe reprezentacje wektorowe:
  - kodowanie 1 z n (ang. *one-hot encoding*)
  - *bag of words (BoW)*
  - n-gramy
  - TF-IDF
- reprezentacje rozproszone:
  - zagnieżdżenia słów (ang. *word embeddings*)
  - wykraczające poza słowa

Dlaczego warto reprezentować wyrazy jako wektory?

- Algorytmy uczenia maszynowego często opierają się na operacjach mnożenia i dodawania macierzy liczb
- Chcielibyśmy wytrenować metody, które będą w stanie rozpoznawać i przewidywać:
  - kontekst
  - relacje między wyrazami
  - analogie
  - wieloznaczność wyrazów
  - synonimy

Wektorowa reprezentacja: kodowanie 1 z n

Kodowanie 1 z n (*one-hot*):

- każdy wyraz reprezentowany jako wektor ($\mathbb{R}^{|V|\times 1}$), tj. wektor z jedną "1" i wieloma "0" :
- $|V|$ rozmiar wektora jest równy rozmiarowi słownika V

V = [ananas jabłko kompot kotka ma pomarańcza pies sok]

kotka (3) [0 0 0 1 0 0 0 0]

pies (6) [0 0 0 0 0 0 1 0]

Wektorowa reprezentacja: kodowanie 1 z n

Problem:

kotka [0 0 0 1 0 0 0 0 0] i

pies [0 0 0 0 0 0 0 1 0] = 0

Wektorowa reprezentacja dystrybucyjna

Wektorowa semantyka: motywacja

- wyrazy o podobnym znaczeniu występują w tekście w podobnych kontekstach
- Firth (1957): "*You shall know a word by the company it keeps*"

Przykład (Nida (1975), Lin (1998), Jurafsky & Martin (2015)):

a. A bottle of *tesgüino* is on the table.
b. Everybody likes *tesgüino*.
c. *Tesgüino* makes you drunk.
d. We make *tesgüino* out of corn.

Wektorowa reprezentacja dystrybucyjna

Reprezentacja za pomocą cech

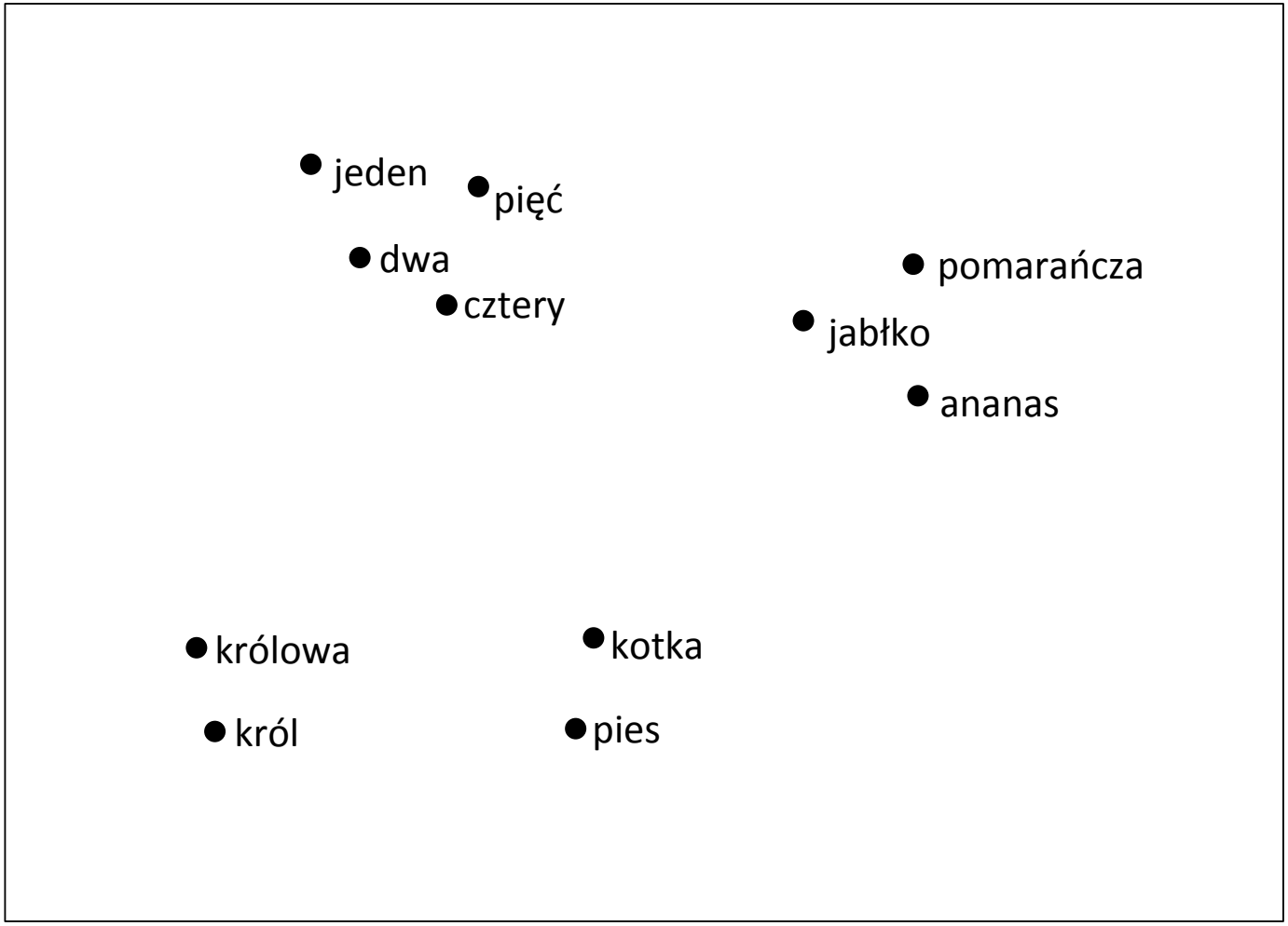
	ananas	jabłko	kompot	kotka	ma	pomarańcza	pies	sok
płeć	0.00	0. 01	0.01	1	0.0	0.01	-1	0.01
jedzenie	0.9	0.98	0.88	0.03	0.0	0.99	0.05	0.91
czasownik	-1	-1	-1	-1	1	-1	-1	-1
płyn	0.2	0.01	0.99	0.01	0.0	0.05	0.01	0.98

Wizualizacja zagnieżdżeń słów (embeddingów)

Możemy dokonać np. rzutowania z przestrzeni wielowymarowej do dwuwymiarowej (2D)
Różne metody redukcji wymiarów, przy zachowaniu struktury:

- PCA (Principal Component Analysis)
- SNE (Stochastic Neighbour Embedding)
- CCA (Curvilinear Component Analysis)
- t-SNE (t-distributed stochastic neighbour embedding)

Wizualizacja zagnieżdżeń słów (embeddingów)



Cztery rodzaje modeli wektorowych

- rzadkie reprezentacje wektorowe:
  - macierze ważonych współwystąpień wyrazów
- gęste reprezentacje wektorowe:
  - rozkład według wartości osobliwych (rozkład SVD)
  - modele oparte o sieci neuronowe (skip-gram, CBOW) (dzisiejszy wykład)
  - hierarchiczna analiza skupień (*Brown clusters*)

Dlaczego gęste reprezentacje?

- wektory w rzadkich reprezentacjach są:
  - długie (długość $|V| = 20\,000$ - $50\,000$)
  - rzadkie (większość elementów to 0)
- wektory w gęstych reprezentacjach są:
  - krótkie (długość 50-1000)
  - 'gęste' (większość elementów to nie 0)

Rzadkie versus gęste wektory

- rzadkie wektory mogą być łatwiejsze w użyciu jako cechy w uczeniu maszynowym (trzeba wyuczyć mniej wag)
- gęste wektory mogą *uogólniać* lepiej niż przechowywanie jawnie liczności
  - lepsze w uchwyceniu synonimów: samochód, auto
  - w praktyce działają lepiej

Modele oparte na predykcji

- Nauka *embeddingów* jako część procesu predykcji wyrazów
- Uczenie sieci neuronowej aby przewidzieć sąsiadujące wyrazy (kontekst)

Kontekst wyrazu

Kontekst wyrazu jest zbiorem k otaczających go wyrazów.

Przykład:
wyraz: kot
zdanie: dzisiaj rano zwinny kot wskoczył na płot
$c = 2$
kontekst k: $\{$rano, zwinny, wskoczył, na$\}$

Softmax

Softmax

Neuronowe probabilistyczne modele języka są tradycyjnie uczone z wkorzystaniem zasady największej wiarygodności (*Maximum Likelihood*) aby zmaksymalizować prawdopodobieństwo wystąpienia kolejnego wyrazu, biorąc pod uwagę poprzednie wyrazy.

Softmax

Uogólnienie regresji logistycznej do wielu klas w celu otrzymania prawdopodobieństwa klasy y:

- dla każdej klasy y mamy osobny wektor wag W_y
- wyliczenie prawdopodobieństwa klas:

$$p(y|x) = \frac{e^{W_y x}}{\sum_{c=1}^{C} e^{W_c x}}$$

gdzie: $W \in \mathbb{R}^{C \times d}$

Softmax c.d.

- dla każdej klasy y mamy osobny wektor wag W_y
- aby obliczyć $p(y|x)$ dla każdej klasy liczymy najpierw iloczyn skalarny $W_y x$, tj. bierzemy wiersz o indeksie y z W i mnożymy przez x:

$$W_y x = \sum_{i=1}^{d} W_{yi} x_i = f_y$$

- Obliczamy wszystkie f_c dla $c = 1, \ldots, C$
- Normalizujemy aby otrzymać prawdopodobieństwo:
- wyliczenie prawdopodobieństwa klas:

$$p(y|x) = \frac{e^{f_y}}{\sum_{c=1}^{C} e^{f_c}}$$

Softmax c.d.

$$P(w_i|w_{i-1}, ..., w_{i-n}) = \text{softmax}(\text{score}(w_i, w_{i-1}, ..., w_{i-n}))$$

score - kompatybilność przewidywanego wyrazu w_i z kontekstem $w_{i-1}, ..., w_{i-n}$ (np. liczona za pomocą iloczynu skalarnego)

Model języka: word2vec

Model języka: word2vec

Mikolov i inni, 2013

- word2vec to grupa powiązanych modeli, używanych do tworzenia (wektorów dystrybucyjnych) *embeddingów*
- modele są płytkimi, dwuwarstwowymi sieciami neuronowymi, trenowanymi do rekonstrukcji kontekstów wyrazów
- kod źródłowy dostępny w sieci
- dwie odmiany: Skip-gram i CBOW

Model języka: word2vec c.d.

Główna idea:

- zamiast *zliczać* (*jak często dany wyraz* w *występuje obok wyrazu 'kot'*)
- wytrenuj prostą sieć neuronową w celu binarnej *predykcji* (*Czy* w *prawdopodobnie pojawi się w pobliżu 'kot'?*)
- sieć z pojedynczą ukrytą warstwą,
- aby wykonać określone zadanie, ale
- nie użyj sieci do zadania, na którym była trenowana, zamiast tego celem jest wyliczenie wag ukrytej warstwy, tj. wektorów wyrazów, których próbujemy się nauczyć (*embeddings*).

Model języka: word2vec c.d.

Genialny pomysł: użyj tekstu jako niejawnie nadzorowanych danych treningowych!

- wyraz w pobliżu 'kot'
- jest przykładem na „prawidłową odpowiedź" na pytanie *Czy* w *prawdopodobnie pojawi się w pobliżu 'kot'?*
- brak potrzeby ręcznego nadzoru

Model Skip-Gram

Skip-Gram

Skip-Gram jest modelem, który pozwala na predykcję kontekstu otaczającego centralny wyraz.

Przykład:
centralny wyraz: skoczył
predykcja/generacja kontekstu: {zwinny, kot, nad, kałużą}

Model Skip-Gram: algorytm

1. Traktuj słowo docelowe i słowo z sąsiedniego kontekstu jako pozytywne przykłady
2. Wylosuj inne słowa z leksykonu, aby uzyskać negatywne próbki
3. Użyj regresji logistycznej aby wytrenować klasyfikator w celu odróżnienia tych dwóch przypadków
4. Użyj wag jako *embeddingi*

Model Skip-Gram: dane trenujące

dzisiaj	rano	zwinny	kot	wskoczył	na	płot
		c1	c2	docelowe	c3	c4

Model Skip-Gram: cel

Mając parę (t, k), gdzie t - słowo docelowe, k - kontekst

- (wskoczył, na)
- (wskoczył, borówki)

Zwróć prawdopodobieństwo tego, że k jest naprawdę wyrazem z kontekstu

$P(+|t, k)$

$P(-|t, k) = 1 - P(+|t, k)$

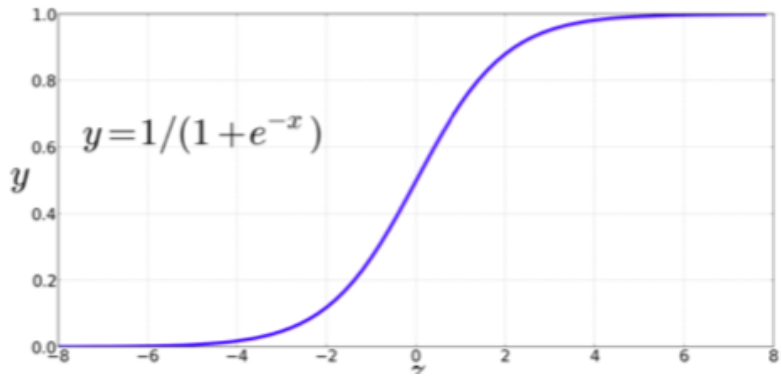
Jak obliczyć $p(+|t,k)$

Pewne intuicje:

- wyrazy prawdopodonie występują obok podobnych wyrazów
- zamodeluj prawdopodobieństwo za pomocą iloczynu skalarnego $t \cdot k$
- problem: iloczyn skalarny nie jest prawdopodobieństwem

Zamiana iloczynu skalarnego w prawdopodobieństwa

$$\sigma = \frac{1}{1+e^{-x}}$$



Zamiana iloczynu skalarnego w prawdopodobieństwa c.d.

$$P(+|t,k) = \frac{1}{1+e^{-t\cdot k}}$$
$$P(-|t,k) = 1 - P(+|t,k) = \frac{e^{-t\cdot k}}{1+e^{-t\cdot k}}$$

Dla wszystkich wyrazów kontekstu

Zakładając, że wyrazy kontekstu są niezależne

$P(+|t, k_{1:n}) = \prod_{i=1}^{n} \frac{1}{1+e^{-t \cdot k}}$

$\log P(+|t, k_{1:n}) = \sum_{i=1}^{n} log \frac{1}{1+e^{-t \cdot k}}$

Model Skip-Gram: dane trenujące

Zdanie trenujące:

dzisiaj	rano	zwinny	kot	wskoczył	na	płot	Dane
		c1	c2	t	c3	c4	

trenujące: pary wejście/wyjście, skupiając się na wskoczył

Załóżmy okno +/- 2 wyrazy

Model Skip-Gram: dane trenujące

Zdanie trenujące:

dzisiaj	rano	zwinny	kot	wskoczył	na	płot
		c1	c2	t	c3	c4

Przykłady pozytywne +

t	k
wskoczył	zwinny
wskoczył	kot

Dla każdego pozytywnego przykładu utworzymy n negatywnych przykładów, używając dowolnych wyrazów, które nie są t

Model Skip-Gram: dane trenujące

Zdanie trenujące:

dzisiaj	rano	zwinny	kot	wskoczył	na	płot
		c1	c2	t	c3	c4

Przykłady negatywne -

t	k
wskoczył	borówki
wskoczył	chmurka

Dla każdego pozytywnego przykładu utworzymy n negatywnych przykładów, używając dowolnych wyrazów, które nie są t

Setup

Przedstawmy wyrazy jako wektory o pewnej długości (standardowo 300) inicjowane losowo.
Zaczynamy od 300 * V losowych parametrów
Na całym zbiorze trenującym chcielibyśmy dostosować te wektory wyrazów tak, że

- maksymalizujemy podobieństwo wyrazów docelowych i par z kontekstu (t, k) z pozytywnych danych
- minimalizujemy podobieństwo par (t, k) z danych negatywnych

Nauka klasyfikatora

Proces iteracyjny.
Zaczniemy od 0 lub losowych wag
Następnie dostosowujemy wagi wyrazów, aby:

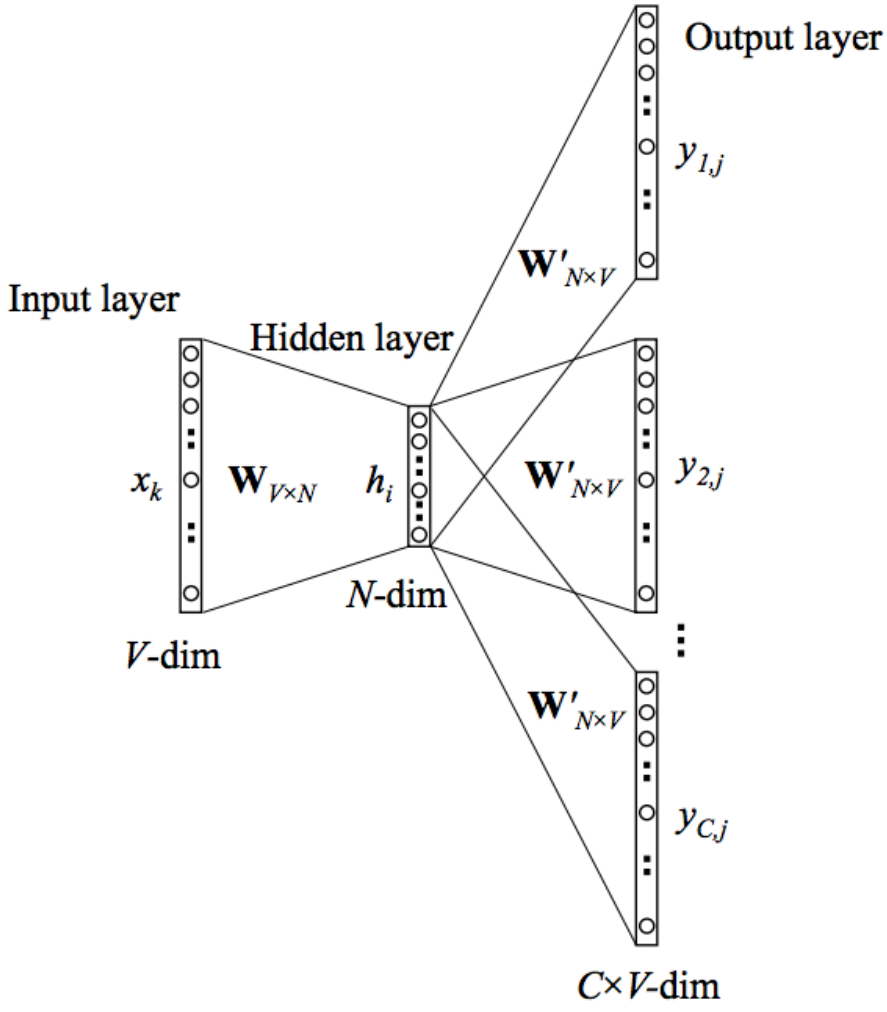
- pozytywne pary były bardziej prawdopodobne
- negatywne pary były mniej prawdopodobne

na całym zbiorze trenującym

Model Skip-Gram: bardziej formalna notacja

- w_i: wyraz i ze słownika V
- $\mathcal{V} \in \mathbb{R}^{n \times |V|}$: wejściowa macierz wyrazów
- v_i: i-ta kolumna $\mathcal{V}$, tj. reprezentacja wyrazu w_i jako n-wymiarowy wektor wejściowy
- $\mathcal{U} \in \mathbb{R}^{n \times |V|}$: wyjściowa macierz wyrazów
- u_j: j-ty wiersz $\mathcal{U}$, reprezentacja wyrazu w_j jako n-wymiarowy wektor wyjściowy
- n: rozmiar przestrzeni *embeddingów*

Model Skip-Gram: architektura



https://commons.wikimedia.org/wiki/File:Skip-gram.png

Model Skip-Gram: funkcja kosztu

minimalizuj $J = -logP(w_{c-m}, \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+m}|w_c)$
Uczenie za pomocą algorytmu spadku gradientu.

Model Skip-Gram: intuicja c.d.

- podczas testowania danego wyrazu dostajemy wynik w postaci listy najbardziej prawdopodobnych sąsiadów
- dzięki *softmax*, dostajemy jeden z wyrazów
- warstwa ukryta pełni rolę "lookup table": mnożenie wektora 1 z n służy do selekcji z macierzy wag jednego rzędu

Podsumowanie: jak wyuczyć *embeddingi* word2vec (Skip-Gram)

Zacznij z V losowymi 300-wymiarowymi wektorami jako początkowe *embeddingi*
Użyj regresji logistycznej

- Weź korpus i weź pary wyrazów, które współwystępują jako pozytywne przykłady
- Weź pary wyrzów, które nie współwystępują jako przykłady negatywne
- Trenuj klasyfikator, aby je rozróżniał, powoli dostosowując wszystkie *embeddingi*, aby poprawić efektywność klasyfikatora
- Wyrzuć kod klasyfikatora i zostaw *embeddingi*

Model CBOW

CBOW

CBOW (*continuous bag-of-words*) jest modelem, który pozwala na predykcję centralnego wyrazu z otaczającego go kontekstu.

Przykład:
kontekst: {zwinny, kot, nad, kałużą}
predykcja/generacja centralnego wyrazu: skoczył

Model CBOW: notacja

taka sama jak dla Skip-Gram

Model CBOW: uczenie

1. Generujemy wektory *1 z n* $(x^{(c-m)}, \ldots, x^{(c-1)}, x^{(c+1)}, \ldots, x^{(c+m)})$ dla kontekstu wejściowego o rozmiarze m.
2. Otrzymujemy dystrybucyjne wektory wyrazów dla kontekstu $(v_{c-m} = \mathcal{V}x^{(c-m)}, v_{c-m+1} = \mathcal{V}x^{(c-m+1)}, \ldots, v_{c+m} = \mathcal{V}x^{(c+m)})$
3. Uśredniamy wektory aby otrzymać $\hat{v} = \frac{v_{c-m}+v_{c-m+1}+\ldots+v_{c+m}}{2m}$
4. Generujemy wektor wynikowy $z = \mathcal{U}\hat{v}$
5. Przekształacamy wyniki w prawdopodobieństwa: $\hat{y} = \mathrm{softmax}(z)$
6. Chcemy aby prawdopodobieństwa $\hat{y}$ były dopasowane do prawdziwych prawdopodobieństw y, co jest wektorem *1 z n* wyrazu.

Model CBOW: funkcja kosztu

cross entropy:

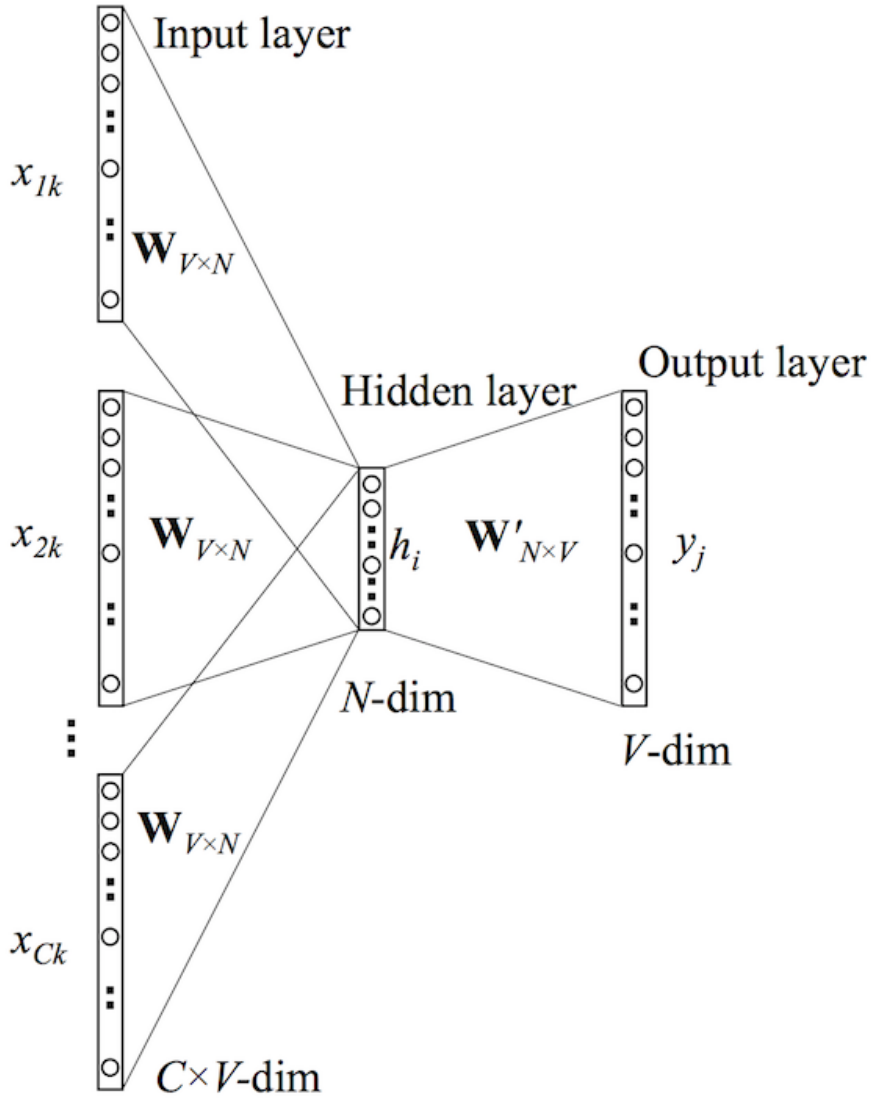
$$H(\hat{y}, y) = -\sum_{j=1}^{|V|} y_j \log(\hat{y}_j)$$

Dla y jako wektora 1 z n:

$$H(\hat{y}, y) = -y_i \log(\hat{y}_j)$$

minimalizuj $J = -logP(w_c|w_{c-m}, \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+m})$

Model CBOW: architektura



https://commons.wikimedia.org/wiki/File:Cbow.png

Model CBOW: intuicja

- konkretny wyraz jest wejściem do uczenia sieci tyle razy, ile ma sąsiadów w kontekście
- trenujemy w ten sposób wszystkie wyrazy
- w rezultacie, jeśli mamy 300 neuronów w warstwie ukrytej to uzyskamy 300 wag, które ustanowią wektor dystrybucyjny tego wyrazu
- wymiar wektora wektora dystrybucyjnego (np. 300) jest liczbą cech, których się wyuczyliśmy

Inne modele

- GloVe (Pennington, Socher, Manning, 2014)
- fastText (biblioteka stworzona przez AI Research (FAIR) Facebooka)

GloVe

- nienadzorowany algorytm uczenia się w celu uzyskanie reprezentacji wektorów dla wyrazów.
- uczenie jest wykonywane na zagregowanych, globalnych statystykach współwystąpień wyrazów w korpusie, a uzyskane w ten sposób reprezentacje przedstawiają liniowe podstruktury przestrzeni wektorowej wyrazów

GloVe - najbliżsi sąsiedzi

3. litoria

4. leptodactylidae

5. rana

7. eleutherodactylus

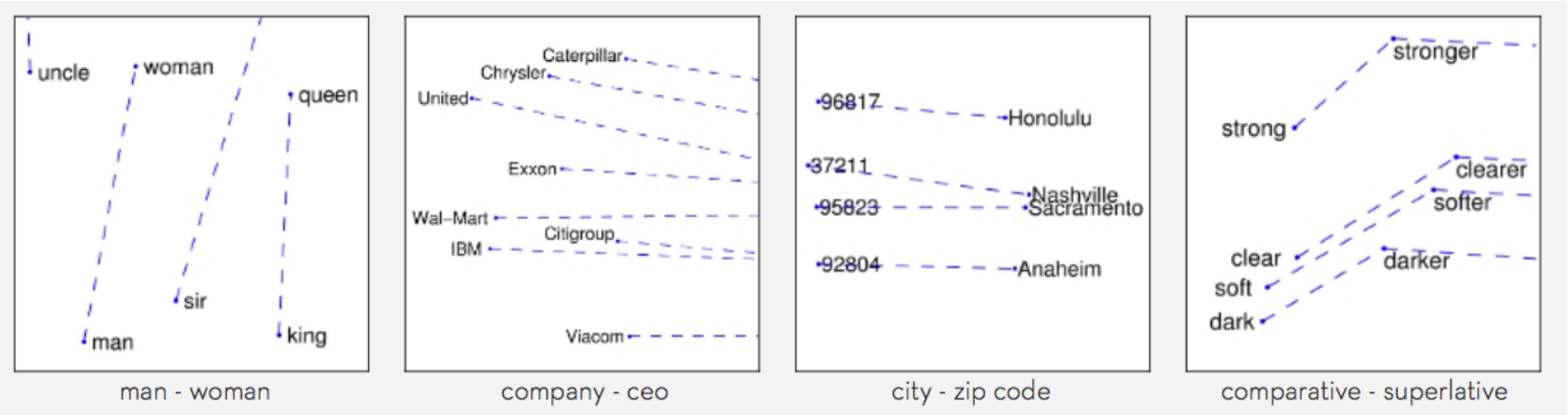
Przykład - najbliższe wyrazy dla wyrazu frog:

1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

- Odległość euklidesowa (lub podobieństwo kosinusowe) między dwoma wektorami wyrazów zapewnia skuteczną metodę pomiaru podobieństwa językowego lub semantycznego odpowiednich wyrazów
- Czasami najbliżsi sąsiedzi według tej metryki ujawniają rzadkie, ale istotne słowa, które leżą poza przeciętnym ludzkim słownictwem

Przykład pochodzi z: `https://nlp.stanford.edu/projects/glove/`

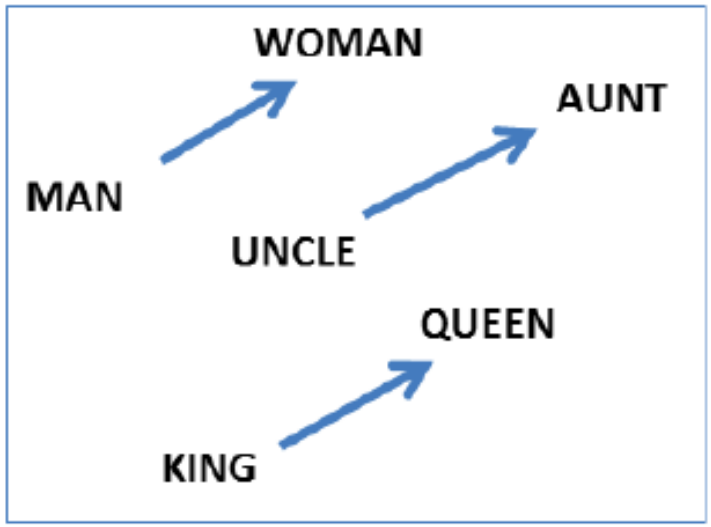
GloVe - liniowe podstruktury

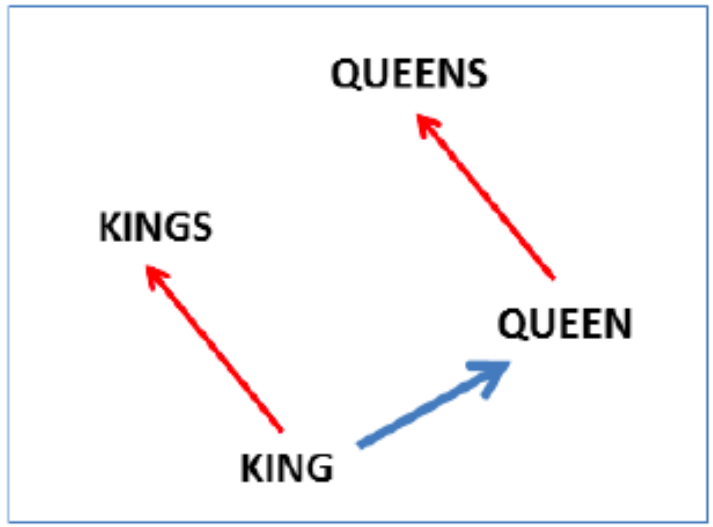


Przykład pochodzi z: `https://nlp.stanford.edu/projects/glove/`

Własności zagnieżdżeń słów (embeddingów)

Relacje pomiędzy wyrazami - analogie





offsety pomiędzy wektorami mogą uchwycić relacje pomiędzy wyrazami, np.:

wektor('king')-wektor('man')+wektor('woman') $\approx$ wektor('queen')

Mikolov, T.; Yih, W.-t. & Zweig, G. (2013), Linguistic Regularities in Continuous Space Word Representations., in 'HLT-NAACL' , pp. 746–751.

Podobieństwo jest zależne od rozmiaru okna c

$c = 2$ Najbliższe wyrazy do *Hogwarts*: *Sunnydale*, *Evernight*
$c = 5$ Najbliższe wyrazy do *Hogwarts*: *Dumbledore*, *Malfoy*, *halfblood*

Transfer learning

Wyuczone na jednym (większym) korpusie zagnieżdżenia słów *embeddingi* można "dotrenować" i zastosować na innym, mniejszym i np. bardziej specjalistycznym.

Embeddingi i stronniczość (*bias*)

Embeddingi odzwierciedlają kulturowe stereotypy

Bolukbasi, Tolga, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam T. Kalai. "Man is to computer programmer as woman is to homemaker? debiasing word embeddings." In Advances in Neural Information Processing Systems, pp. 4349-4357. 2016.

"Paris:France :: Tokyo : x"

- x = Japan

"father : doctor :: mother : x"

- x = nurse

"man : computer programmer :: woman : x"

- x = homemaker

Embeddingi odzwierciedlają kulturowe stereotypy

Caliskan, Aylin, Joanna J. Bruson and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. Science 356:6334, 183-186.

- Afroamerykańskie nazwy kojarzą się z nieprzyjemnymi wyrazami (więcej niż nazwy europejsko-amerykańskie)
- Męskie imiona związane bardziej z matematyką, imiona żeńskie ze sztuką
- Imiona starszych ludzi z nieprzyjemnymi wyrazami, młodych ludzi z przyjemnymi wyrazami.

Embeddingi odzwierciedlają i replikują różne rodzaje szkodliwych uprzedzeń.

Linki do pobrania *embeddingów*

Word2vec (Mikolov et al.)
`https://code.google.com/archive/p/word2vec/`
Fasttext `http://www.fasttext.cc/`
Glove (Pennington, Socher, Manning)
`http://nlp.stanford.edu/projects/glove/`

Dziękuję za uwagę!