TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Vektorová reprezentace slov

Word embeddings

Petr Červa, František Kynych

26. 9. 2024 | MVD

Část I.:
Úvod do problematiky

Význam slova a jeho zakódování

- Význam slova odpovídá jeho informačnímu obsahu
- Jak význam zakódovat užitečně pro počítač?
  - Klasické řešení: pomocí taxonomie (sítě) popisující vztahy a podobnost mezi slovy
    - WordNet
    - Celá řada problémů
      - Subjektivní
      - Vyžaduje velké množství ruční práce
      - Jak udržovat aktuální?

=> většina přístupů nakonec pracovala se slovy pouze jako s diskrétními objekty

Diskrétní reprezentace slov

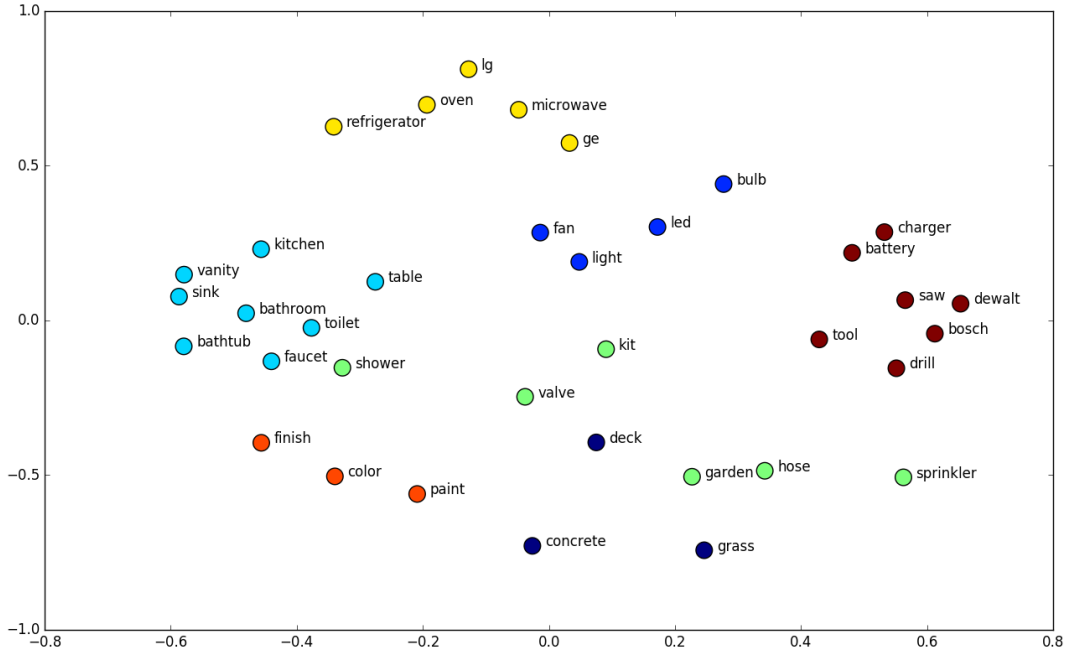
- Každé slovo reprezentováno samostatně
- Pro daný slovník zakódováno jako vektor nul s jednou jedničkou
  - „One-hot encoding“:

    Petr = [0 0 … 0 1 0 … 0 0]

    Pavel = [0 1 0 …0 ….0]
  - Celá řada problémů
    - Prostor má obří dimenzi |V|
    - Poloha vektorů a vzdálenosti mezi nimi nijak nesouvisí s významem slova
    - Podobná slova neleží ve stejné oblasti prostoru

Reprezentace přes podobnostní rozložení

- Cílem je vytvořit matematický model reprezentující podobnostní rozložení slov -> najít prostor, v němž jsou slova rozložena podle významu
- Každé slovo bude v tomto prostoru reprezentováno vektorem, který bude ležet blízko vektorů slov s podobným významem



Jak podobnostní rozložení získat?

- Distribuční hypotéza
  - Klíčem je myšlenka, že význam slova je dán okolím slova
  - Slova se stejným významem mají podobné okolí

„You shall know the word by the company it keeps“ (Firth, J. R. 1957:11)

- Prostor hledáme tak, že daný model maximalizuje pro všechna slova pravděpodobnost výskytu jejich okolí

$$P(kontext|wt)$$

Metody hledání podobnostního rozložení

- Historicky více přístupů:
  - Learning representations by back-propagating errors
    (David E. Rumelhart, Geoffrey E. Hinton & Ronald J. Williams, 1986)
  - A neural probabilistic language model
    (Bengio et al., 2003)
  - …
  - **Word2Vec**
    (Mikolov et al., 2013)
  - **GloVe**
    (Pennington et al., 2014)

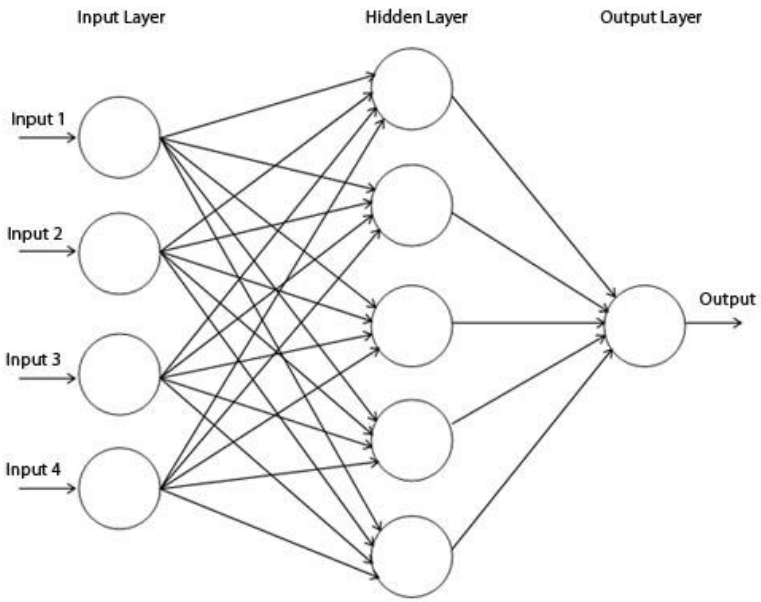
Word2Vec

- Využívá neuronovou síť s jednou skrytou vrstvou
- Dva možné algoritmy
  - **Skip-gram**
    - Založen na predikce slov v okolí daného slova
  - Continuous bag of words (**CBOW**)
    - Založen na predikci daného slova pomocí okolních slov
- Několi trénovacích metod
  - Softmax
  - Hierarchical Softmax
  - Negative sampling

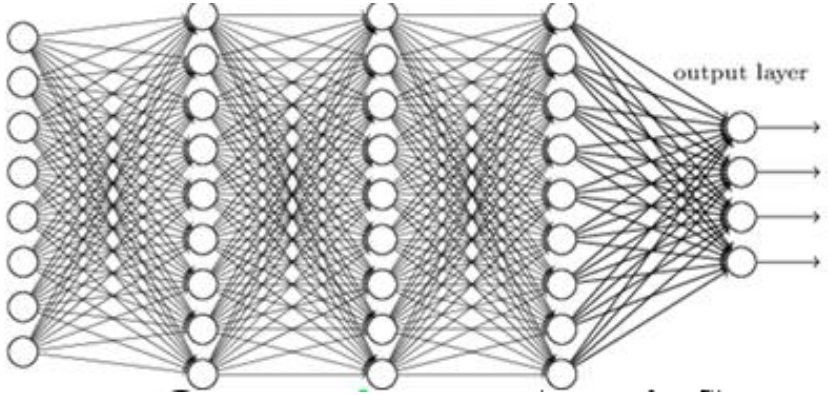
Část II.:
Opakování z předmětu USU

Co je to umělá neuronová síť?

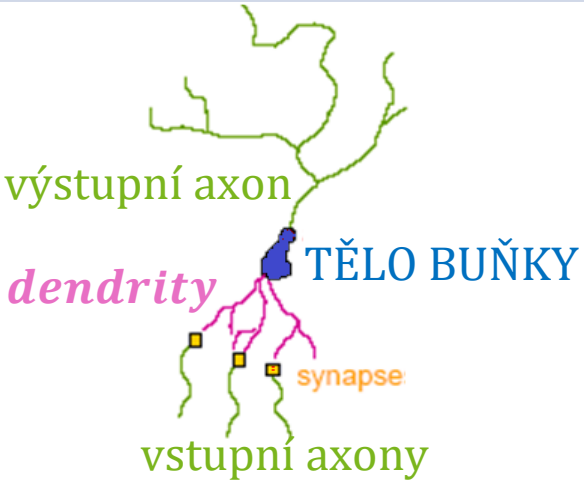
- Matematický model tvořený sériovým a paralelním spojením umělých neuronů
- Podle typů spojení a uspořádání neuronů existuje celá řada typů umělých neuronových sítí
  - Následující výklad bude omezen na architekturu označovanou jako **Multi-Layer Perceptron (MLP),** kde jsou
    1. Neurony uspořádány vedle sebe do vrstev a vrstvy jsou spojeny sériově za sebou
    2. Výstup z každého neuronu vrstvy i je přiveden na vstup všech neuronů vrstvy $i+1$
- Inspirací je mozek a biologické neuronové sítě

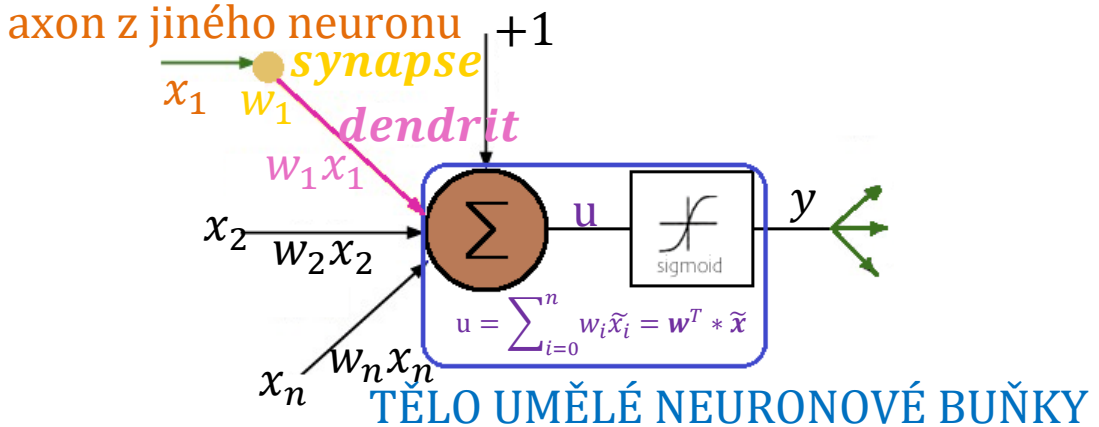


Mozek	Umělá neuronová síť typu MLP
Skládá se z asi 10^{11} buněk - neuronů	Skládá se z několika až několika desítek tisíc umělých neuronů
Neurony jsou různě nepravidelně pospojovány do sítě	Umělé neurony jsou pospojovány paralelně do vrstev, a vrstvy jsou pak spojeny mezi sebou sériově = pravidelná struktura
Proces učení spočívá v aktivování a zesilování/zeslabování synapsí (vazeb) mezi r	Proces učení spočívá v nastavení a následném neustálém přepočítávání váhových koeficientů w na každém vstupu do každého neuronu



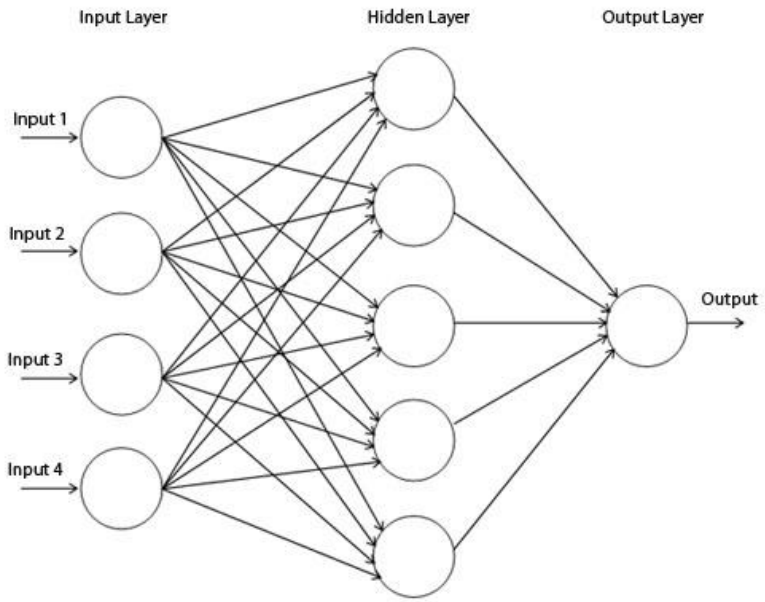
Neurony	Umělé neurony
Spoje mezi neurony tvoří axony	Spoje jsou dány logickou strukturou sítě, kdy výstup z jedné vrstvy/vstupní data je naveden na vstupy do další/první vrstvy
Spoj je tvořen synapsí = parametr učení	Parametrem učení je vektor vah w
Samotný vstup do buňky tvoří dendrity	Dendrit lze chápat jako hodnotu součinu $w_i x_i$
Podněty jsou v těle neuronu akumulovány	Vstupní signály jsou v modelu sčítány
Po překročení určitého prahu je podnět poslán dál	Po překročení prahové hodnoty je výstupní signál změněn podle typu použité aktivační funkce





Síť typu vícevrstvý perceptron - architektura

- Obecně obsahuje vícevrstvý perceptron
  - **Vstupní vrstvu**: nemá neurony, reprezentuje vektor vstupních hodnot
  - Jednu nebo více **skrytých vrstev** s aktivační funkcí
  - **Výstupní vrstvu**
- Výstupní vrstva a skryté vrstvy mají vždy vlastní matice váhových koeficientů **W**



Funkce SOFTMAX

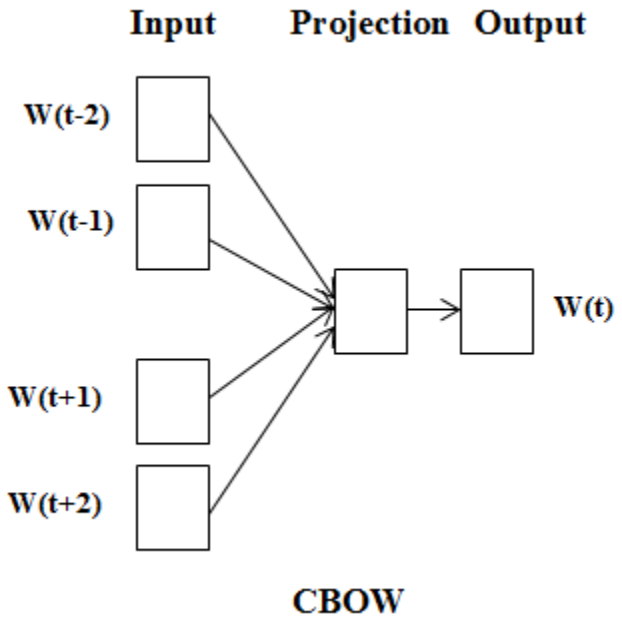
- Funkce SOFTMAX má C vstupů a C výstupů
- Platí, že výstup

$$\hat{y}_c = SOFTMAX(\boldsymbol{u}) = \frac{e^{u_c}}{\sum_{d=1}^{C} e^{u_d}}$$

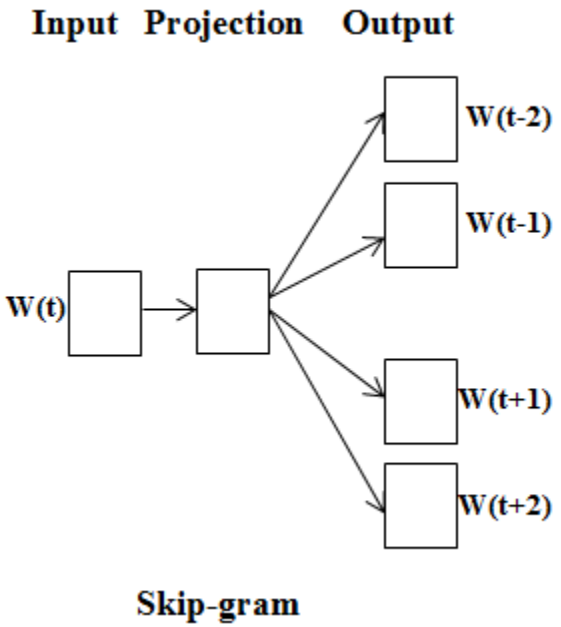
  - Všechny výstupy jsou kladná čísla
  - Součet všech výstupů dohromady je roven číslu 1

Část III.:
Word2Vec

Přístup: CBOW vs Skip-gram



CBOW



Skip-gram

Skip-gram princip

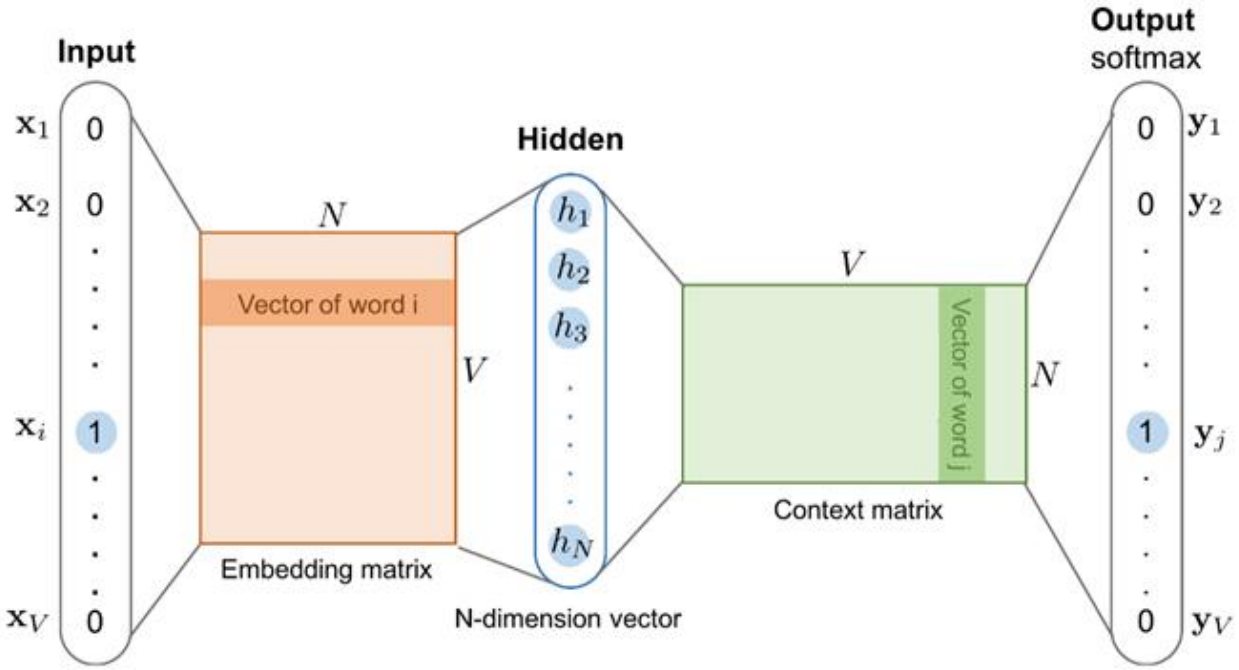
- Maximalizujme pravděpodobnost výskytu okolních slov
- Příklad
  - Petr dnes šel do kina
  - Při šířce okolí 2 maximalizujeme pro slovo šel pravděpodobnost
    $P(Petr \mid šel, \boldsymbol{W}) + P(dnes \mid šel, \boldsymbol{W}) + P(do \mid šel, \boldsymbol{W}) + P(\text{kina} \mid šel, \boldsymbol{W})$
- Model přitom odpovídá neuronové síti s parametry $\boldsymbol{W}$
  - Parametry $\boldsymbol{W}$ jsou matice vah skryté a výstupní vrstvy ($\boldsymbol{W}_1$ a $\boldsymbol{W}_2$)
    - Matice $\boldsymbol{W}_1$ pak představuje matici embeddingů pro všechna slova ze slovníku
    - Její rozměry (počet neuronů skryté vrstvy) určují dimenzi nalezeného prostoru
    - Nalezený prostor má požadované vlastnosti

Příklad trénování

- Slova ze slovníku zakódujeme pomocí „one-hot“ kódování
- Požadovaný vektor (zde pro slovo „šel“) přivedeme na vstup
- Na výstupu chceme získat postupně vektory odpovídající slovům „Petr“, „dnes“, „do“ a „kina“
- Například pro dvojici „šel“ a „Petr“ je
  - Vstupní vektor x = [0 0 1 0 0] a požadovaný výstupní vektory y = [1 0 0 0 0]
- Výstup ze sítě (funkce softmax) ovšem neodpovídá vždy přesně požadovanému
  - Může být například [0.7 0.1 0.1 0.05 0.05]
- Vznikne chyba, která se poté posčítá za všechna okolní slova
  - Na základě celkové chyby se přepočítají parametry modelu
    - Jde o algoritmus zpětné propagace (viz předmět USU)
    - Během trénování minimalizujeme cross-entropii mezi skutečným výstupem ze sítě a požadovanými hodnotami

Word2vec – schéma modelu

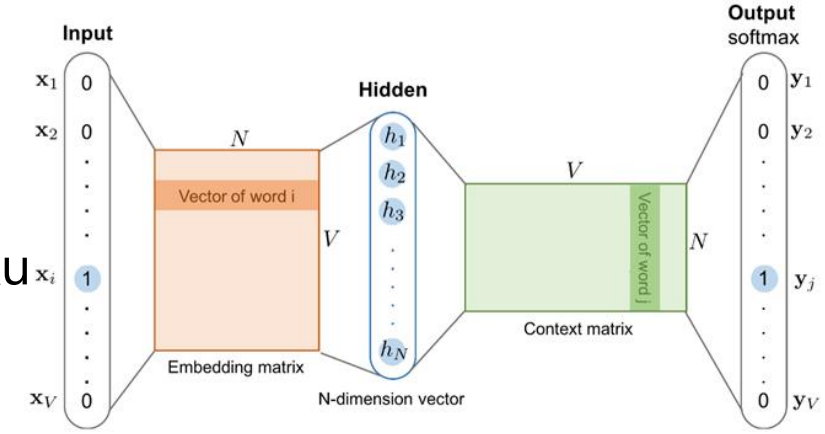
- Jde o model odpovídající NS s jednou skrytou vrstvou
- Parametry modelu jsou matice $\boldsymbol{W}_1$ a $\boldsymbol{W}_2$
- Na výstupu je softmax => krit. funkce má význam křížové entropie



https://towardsdatascience.com/word2vec-made-easy-139a31a4b8ae

Word2vec – schéma #2

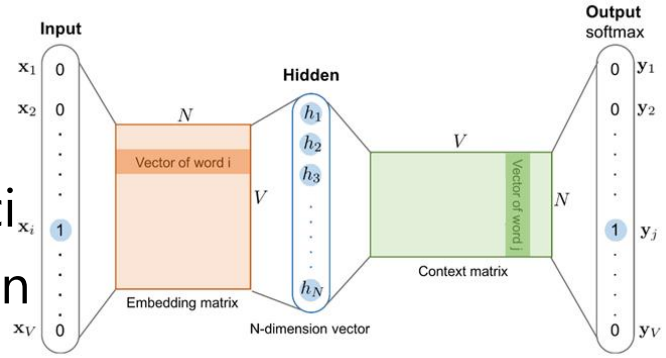
- $\boldsymbol{x}$
  - vektor reprez. vstupní slovo
  - dimenze [1, V]
  - obsahuje pouze jednu jedničku



- $\boldsymbol{W}1$
  - matice vah skryté vrstvy o dimenzi [V, N]
  - **reprezentuje word embedings**
    - na řádku v je embedding pro v-té slovo ze slovníku
    - vektor x obsahuje pouze jednu jedničku
    - Součinem $\boldsymbol{x}^T \boldsymbol{W}_1$ proto vybereme vždy jeden příslušný řádek této matice

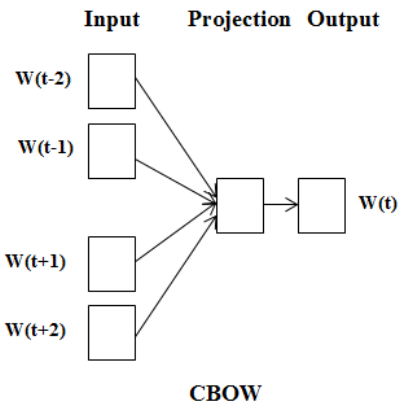
Word2vec – schéma #3

- $\boldsymbol{h}=\boldsymbol{x}^T\,\boldsymbol{W}_1$
  - vektor odpovídající embeddingu slova
  - skrytá vrstva má lineární aktivační funkci
  - tento embedding není nijak modifikován
- $\boldsymbol{W}_2$
  - "matice vah výstupní vrstvy dimenzi [N,V]"
  - odpovídá kontextové matici
- $\boldsymbol{y}=\boldsymbol{softmax}(\boldsymbol{h}^T\,\boldsymbol{W}_2)$
  - výstupní vektor, ideálně přesně odpovídá „one hot“ zakódování požadovaného výstupní slova
  - reálně se liší a podle odchylky se při trénování upraví všechny váhy

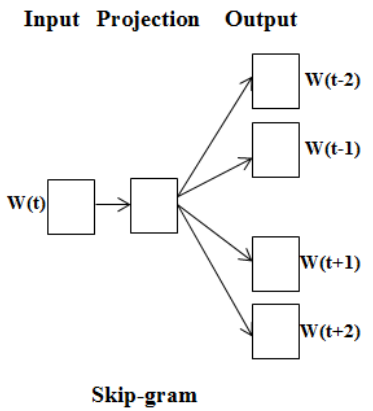


SKIP-GRAM vs CBOW

- U skip-gramu se sečte chyba na výstupu pro všechna predikovaná slova z okolí cílového slova
- U varianty CBOW se maximalizuje pravděpodobnost cílového slova na základě okolních slov
  - Chyba za jednotlivá okolní slova se n výstupu sítě průměruje
- Pro obě metody obvykle platí, že čím větší N tím lepší výsledky
- Okolí se volí cca 10 pro skip-gram a 5 pro CBOW



CBOW

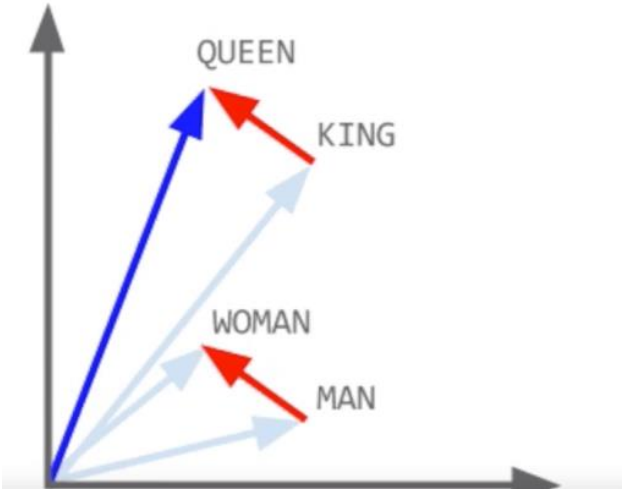


Skip-gram

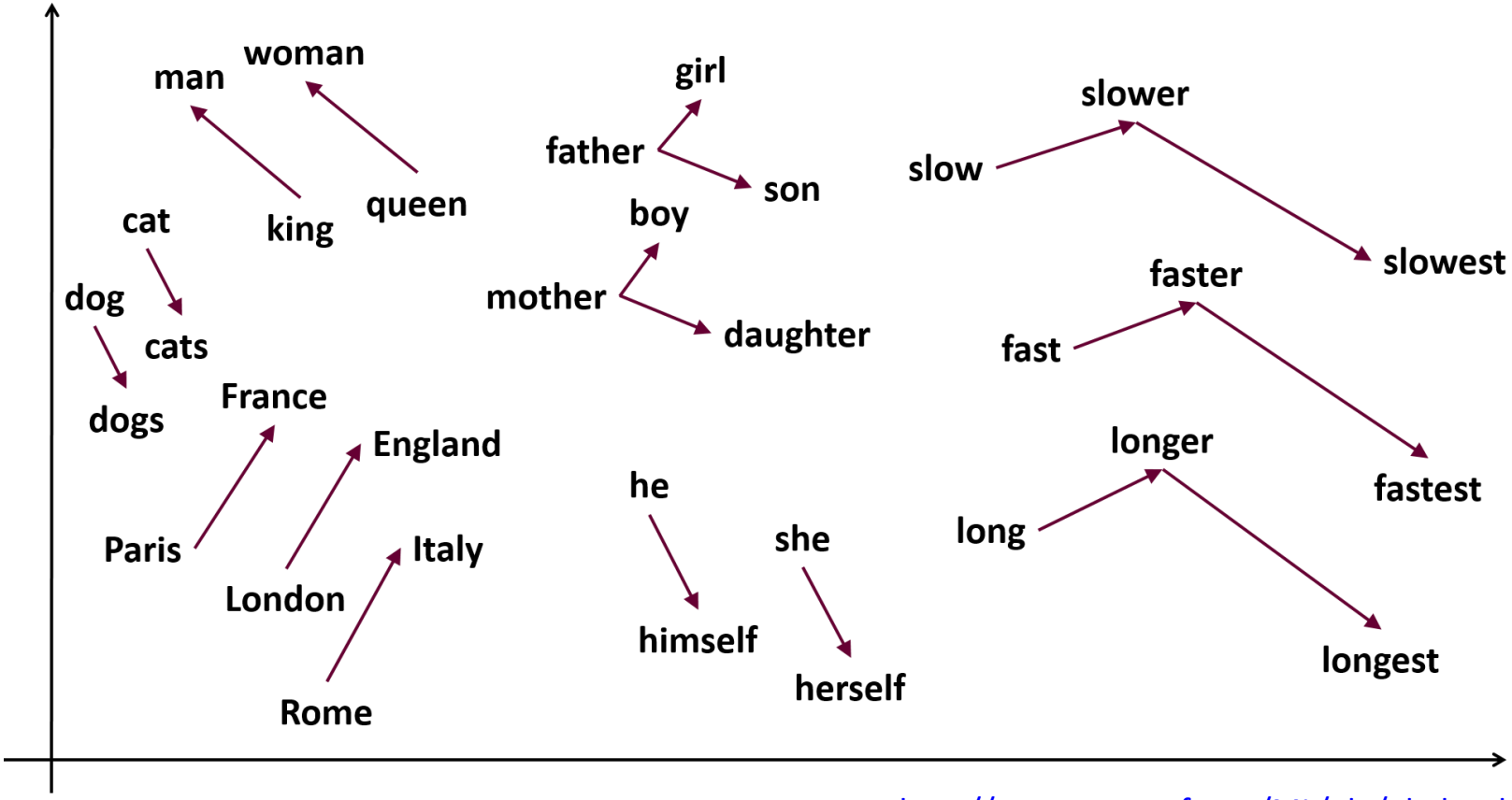
Vlastnosti Word2Vec

- Vektory reprezentující významově podobná slova tvoří shluky
  - Jsou blízko sebe
- Přičítáním a odčítáním vektorů je možné posouvat nebo přenášet význam:

$$v(king) - [v(man) - v(woman)] = v(queen)$$

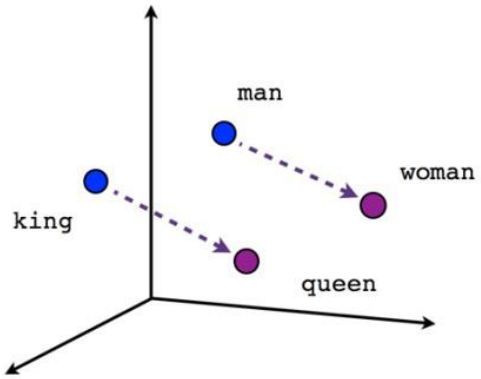


Naučené vztahy mezi slovy

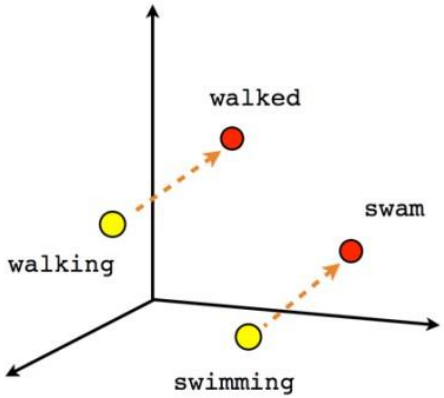


http://www.samyzaf.com/ML/nlp/nlp.html

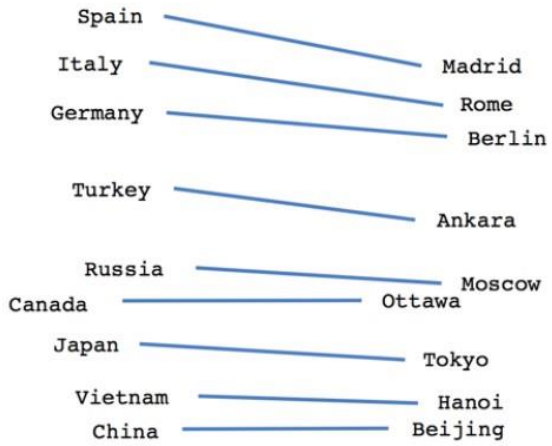
Naučené vztahy mezi slovy



Male-Female



Verb tense



Country-Capital

walking – swimming + swam = walked

walked – walking = swam - swimming

https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

Nevýhody popsaného způsobu trénování

- Výše popsané trénování má nevýhody
  1. Výpočet softmaxu pro velké slovníku je náročný (exponenciála)
  2. Pro dané cílové slovo se významně mění hodnoty pouze omezeného počtu vah, ostatní váhy se přesto aktualizují nadbytečně o zanedbatelně malé hodnoty
- Příklad
  - Předtrénovaný Google model má 3 miliony slov
    - Při trénování je pouze 1 slovo ze 3M cílové
    - Většinu času aktualizujeme váhy, které nesouvisí s cílovým slovem o malé hodnoty (viz 2.)
- Řešení ve výběru trénovací metody
  - Hierarchical softmax
    - Složitější na implementaci
  - **Negative sampling**
    - Dosahuje lepších výsledků

Negative Sampling #1

- Umožňuje pro každý vzorek modifikovat pouze malou část vah
- Problém se reformuluje
  1. Místo predikce slova „šel“ na základě okolních slov se trénuje model predikující, zda je nebo není slovo „šel“ sousední se slovy „Petr“, „do“ …
     - Klasifikaci 1 z V převedeme na V binárních klasifikací
  2. Trénování binárního klasifikátoru pro dané cílové slovo se dále zjednoduší tak, že se kromě slova v okolí cílového slova vybere i několik (typicky 5) náhodně vybraných slov, negativních případů, která v daném okolí zrovna nejsou
     - Dvojici (šel;Petr) doplníme např. o dvojice (šel;dům), (šel;včera)
     - Pro slovo v okolí chceme na výstupu NS hodnotu 1
     - Pro negativní vzorky hodnotu 0

Negative Sampling #2

- Chyba se pak propaguje zpět pouze pro použitá slova a zároveň se nevyčísluje softmax pro V slov

  $\Rightarrow$ zrychlení výpočtu

- Slova se jako negativní vzorky vybírají náhodně s pravděpodobností $P = u(w)^{3/4}$, kde $u(w)$ je unigram daného slova
  - Četnější slova budou vybrána častěji než méně četná
  - Mocnina ¾ zvýhodňuje méně četná slova oproti samotnému unigramovému rozložení

Skip-gram nebo CBOW (a Negative sampling)?

- Skip-gram je pomalejší + lepší pro málo četná slova
- CBOW je rychlejší + lepší pro četná slova
- Softmax je pomalejší + lepší pro málo četná slova
- Negative sampling je rychlejší + lepší pro četná slova a málo dimenzionální vektory

Využití

- Překlad z jazyka do jazyka
- Analýza sentimentu
- Klasifikace textu
- Automatická sumarizace
- Identifikace jazyka z textu
- …

Další metody vektorizace textu

- Klasické starší přístupy
  - Latentní Sémantická Analýza (LSA)
    - Historicky před Word2Vec
    - Pracuje s celým dokumentem a maticí výskytu jednotlivých slov v dokumentu
- Novější
  - GloVe
    - Kombinuje výhody LSA a Word2Vec

GloVe #1

- Global Vectors for Word Representation
- Intuice
  - Word2Vec nepřímo modeluje společný výskyt slov
  - Společný výskyt slov můžeme spočítat přímo v prvním kroku

- 2 možnosti počítání slov
  - Okénko
    - Vybereme n slov okolo slova a v tomto okénku počítáme společný výskyt
    - Zachycuje syntaktickou i sémantickou informaci
  - Dokument
    - Místo počítání v okénku počítáme v celém dokumentu (LSA)

GloVe #2

- Příklad počítání s okénkem
- Text:
  - I like NLP.
  - I like deep learning.
- Okénko: 1 slovo (v okolí)

	I	like	NLP	deep	learning
I	0	2	0	0	0
like	2	0	1	1	0
NLP	0	1	0	0	0
deep	0	1	0	0	1
learning	0	0	0	1	0

GloVe #3

- Problémy takového počítání
  - Matice se zvětšuje s velikostí slovníku
  - V matici se vyskytuje velké množství nul
  - Často vyskytující se slova budou mít velký počet společného výskytu
- Vylepšení
  - Použití Singular Value Decomposition (SVD) pro redukci dimenze
  - Velkou četnost společného výskytu omezíme maximální hodnotou
    - Např. min(100, count)
  - Použití vzdálenosti při počítání v okénku
    - „I like NLP …“
      - X(I, like) += 1
      - X(I, NLP) += $\frac{1}{2}$
      - X(I, …) += $\frac{1}{3}$

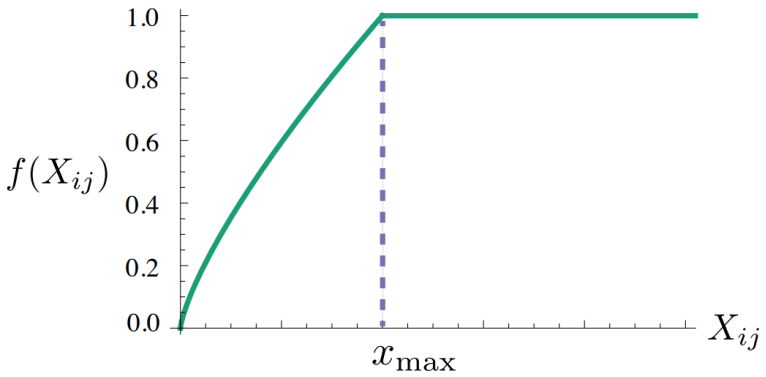
GloVe #4

- Posledním vylepšením je získat každý počet v rozsahu 0-1
  - Lepší pro učení NN

$$f(X_{ij}) = \left(\frac{X_{ij}}{x_{max}}\right)^{\alpha} \text{ if } x < x_{max}, \text{ else } 1$$

α = 0,75; x_{max} = 100

X_{ij} – hodnota z matice počtů společného výskytu



GloVe #5

- SVD se složitěji optimalizuje
  - Problém s časovou náročností pro větší matice

- Výsledné řešení
  - Dvě matice vah *u* a *v* (dimenze [V, N]) a 2× bias vektor
    - Podobně jako u Word2Vec
  - Učí se pomocí váženého MSE

$$J = \frac{1}{2}\sum_{i,j=1}^{V} f(X_{ij})\left(u_i^T v_j + bi + bj - \log X_{ij}\right)^2$$

- Funguje dobře na menším množství dat i s menší velikostí embeddingu

Užitečné odkazy

- Word2Vec na TowardsDataScience
- Stanford University, kurz cs 224n
  - http://web.stanford.edu/class/cs224n/
  - Youtube
- Word2Vec článek
- GloVe článek