Reprezentacje wektorowe tekstu i ich wizualizacja

Machine Learning Gdansk - Hackerspace
20.06.2017

Adam Wróbel

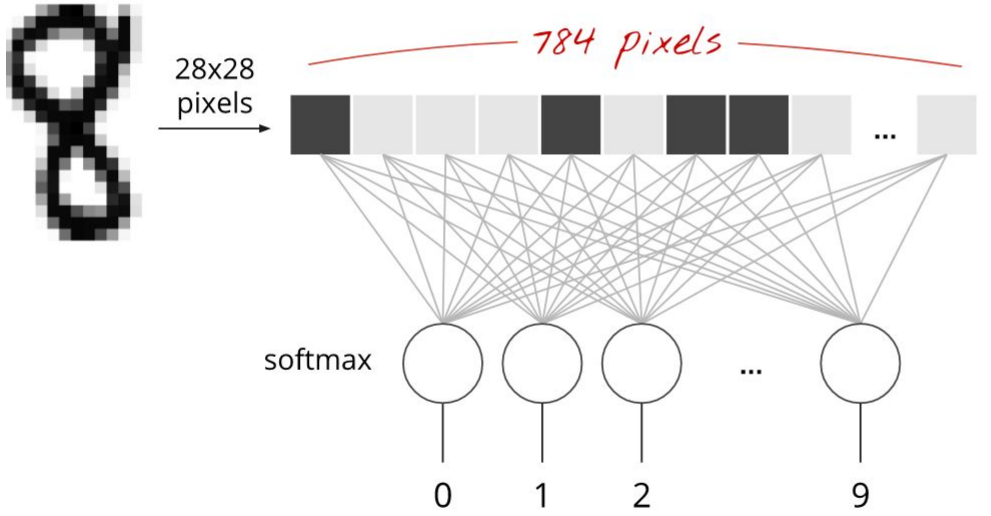
github.com/Adamage
adam.wrobel1@gmail.com
linkedin.com/in/adamwrobel1

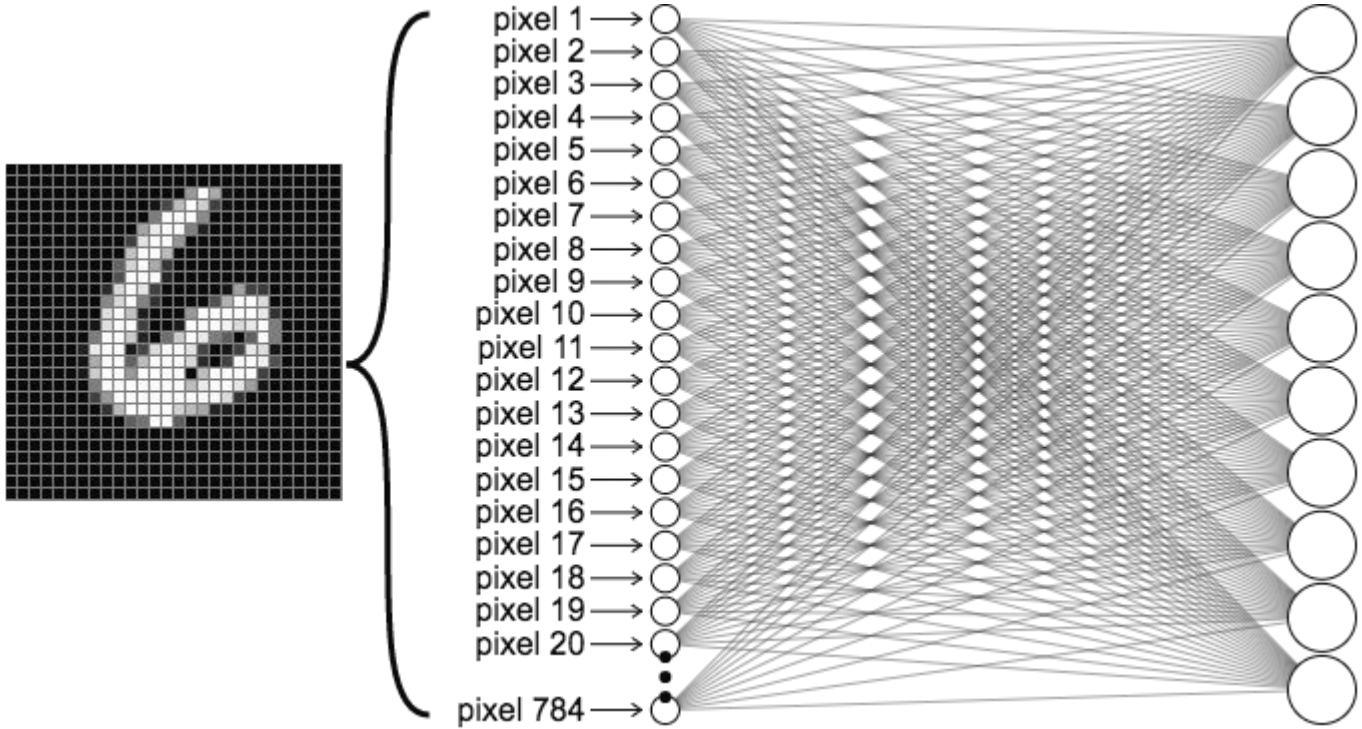
Dlaczego chcemy przedstawić słowa jako wektory?

- Wiekszosc rozwiazan w ML opiera sie na operacjach mnozenia, dodawania, na macierzach liczb

Np. wykrywanie pisanej cyfry
to problem klasyfikacji:
Obrazek 28x28, kazdy piksel
Moze miec wartosc 0 albo 1

Taki wektor musi dac sie
zmapowac do jednej z 10 kategorii
Poprzez np wyuczenie sieci neur.

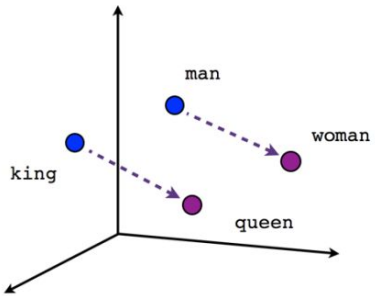




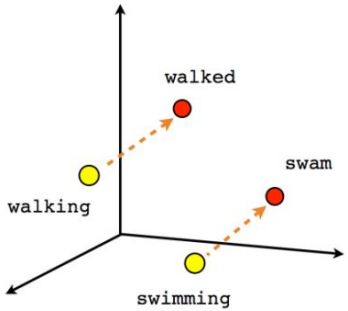
http://ksopyla.com/python/tensorflow/klasyfikacja-cyfr-mnist-tensorflow/

Dlaczego chcemy przedstawić słowa jako wektory?

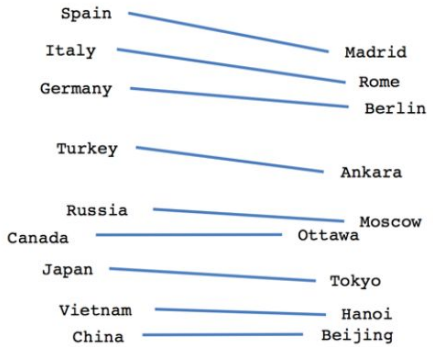
- Potrzebujemy reprezentacji, ktora jest w stanie uchwycic:
  - Wieloznacznosc slow i synonimy
  - Slang, wulgaryzmy
  - Kontekst
  - Relacje miedzy slowami, analogie, np:



Male-Female



Verb tense



Country-Capital

https://www.tensorflow.org/tutorials/word2vec

Jak przedstawić slowa jako wektory?

Intuicja:
Jezeli slowa często występują w podobnym kontekście,
to muszą mieć podobne znaczenie.

Mozemy “nauczyć” model ciągłej dystrybucji prawdopodobieństwa, że dla danego slowa, inne słowa beda w jego kontekscie (zazwyczaj w odleglosci 2-8 słów).

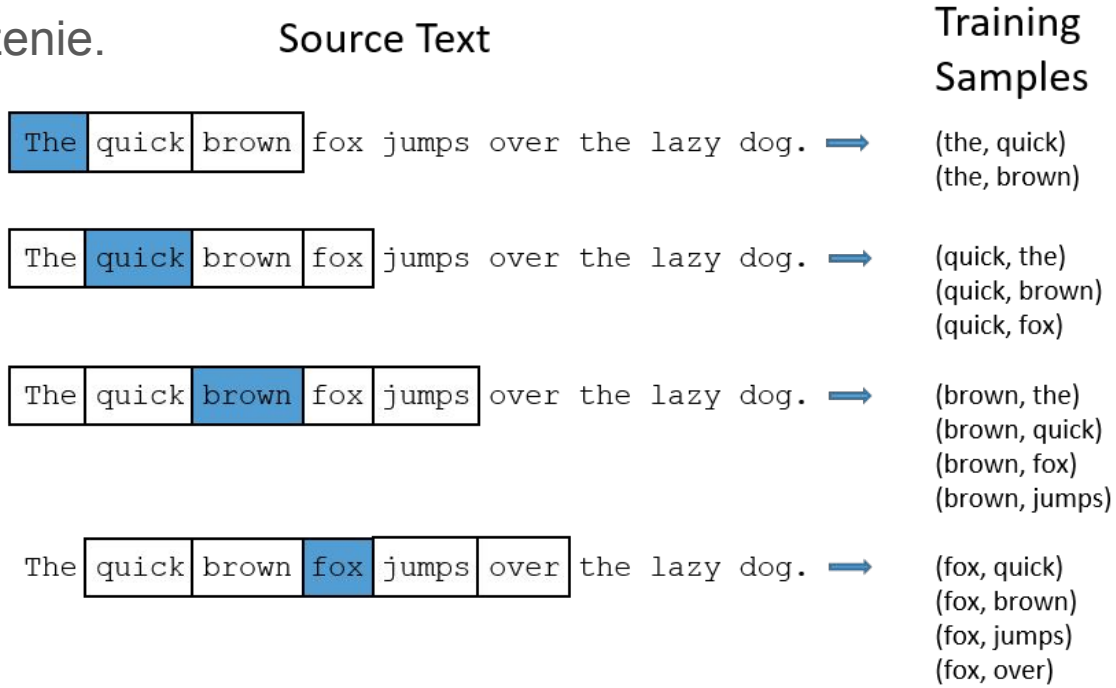
Wyprowadzenie:
http://web.stanford.edu/class/cs224n/lectures/cs224n-2017-lecture2.pdf

Jak przedstawic slowa jako wektory?

Intuicja:
Jezeli slowa często występują w podobnym kontekście, to muszą mieć podobne znaczenie.

Na rysunku:
“ruchome okno”

Wypisujemy wszystkie pary słowa w środku i jego sąsiadów w oknie. Będą to pary input-output do trenowania modelu.



Intuicja odnośnie działania algorytmu

Model **Skip-gram** - predykcja slow w oknie dookola slowa
Model **Continuous Bag of Words** - predykcja slowa posrodku, na podstawie slow w oknie

Uzyc prostej sieci neuronowej, z jedna warstwa ukryta:

- Wejscie to wektor [10000 x 1], z jedna jedynka oznaczajaca TO slowo
- Wyjscie:
    - Przy trenowaniu, to wektor [10000 x 1], z jedna jedynka oznaczajace jednego z sasiadow
    - Przy ewaluacji, wektor [10000 x 1] z rozkladem prawdopodobienstwa, ktore slowa moglyby byc sasiadami

Aplikacja softmaxu na wyjscie z klasyfikacja na jedno z 10000 slow da nam najbardziej prawdopodobnego sasiada.

Trenujemy konkretne slowo na inpucie tyle razy, ile ma sasiadow w oknie, i w ten sposob wszystkie slowa.

W efekcie, jezeli warstwa ukryta ma 300 jednostek, to uzyskamy za pomoca 300 roznych wag, liczby do reprezentacji wektorowej tego slowa!

Wymiar wektora, np. 300, to liczba “cech”, “features”, nauczonych automatycznie.

Intuicja odnosnie dzialania algorytmu

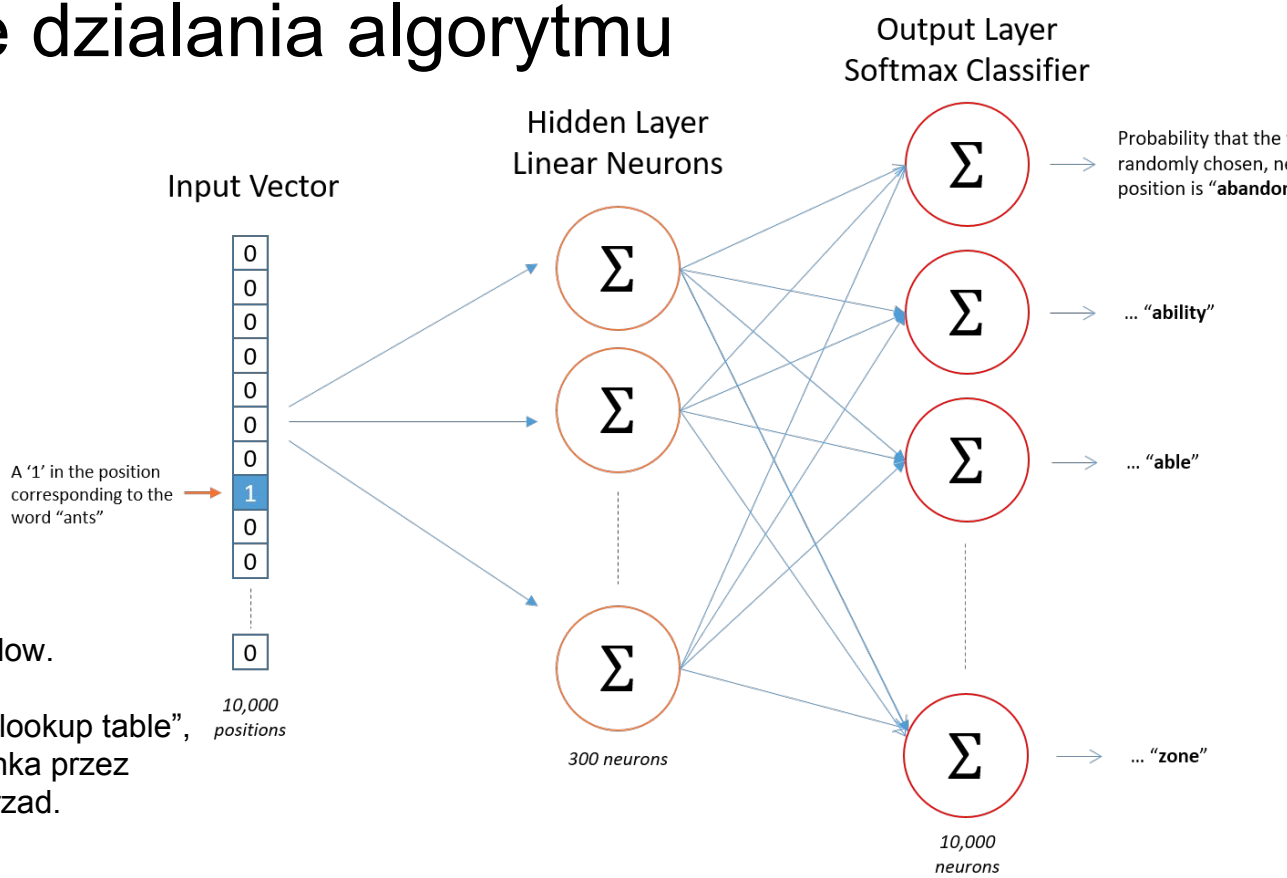
W trakcie trenowania:
- Input to wyraz zakodowany “1”
- Output to wyraz ktory jest na pewno w oknie dookola slowa

Po trenowaniu:
- Wagi 300 neuronow ukrytych sa jednoczesnie wektorem slowa

Przy ewaluacji danego slowa, wynik to wlasciwie lista najbardziej prawdopodobnych sasiadow.

Aplikujac softmax, dostaniemy jedno ze slow.

Tak naprawde warstwa ukryta dziala jak “lookup table”, gdyz przemnozenie wektora z jedna jedynka przez macierz wag “wybiera” z macierzy jeden rzad.



http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Demo: uzycie wytrenowanego modelu

Git clone https://github.com/Adamage/neural-nets-tutorials

https://github.com/Adamage/neural-nets-tutorials/tree/master/tensorflow-tutorials/natural-language/notebook-check-similarities.ipynb

Ten notebook ściągnie dane i wytrenuje model. Następnie możemy podglądać

najbliższe słowa, pod względem częstości występowania w podobnych konteksach.

$ jupyter notebook

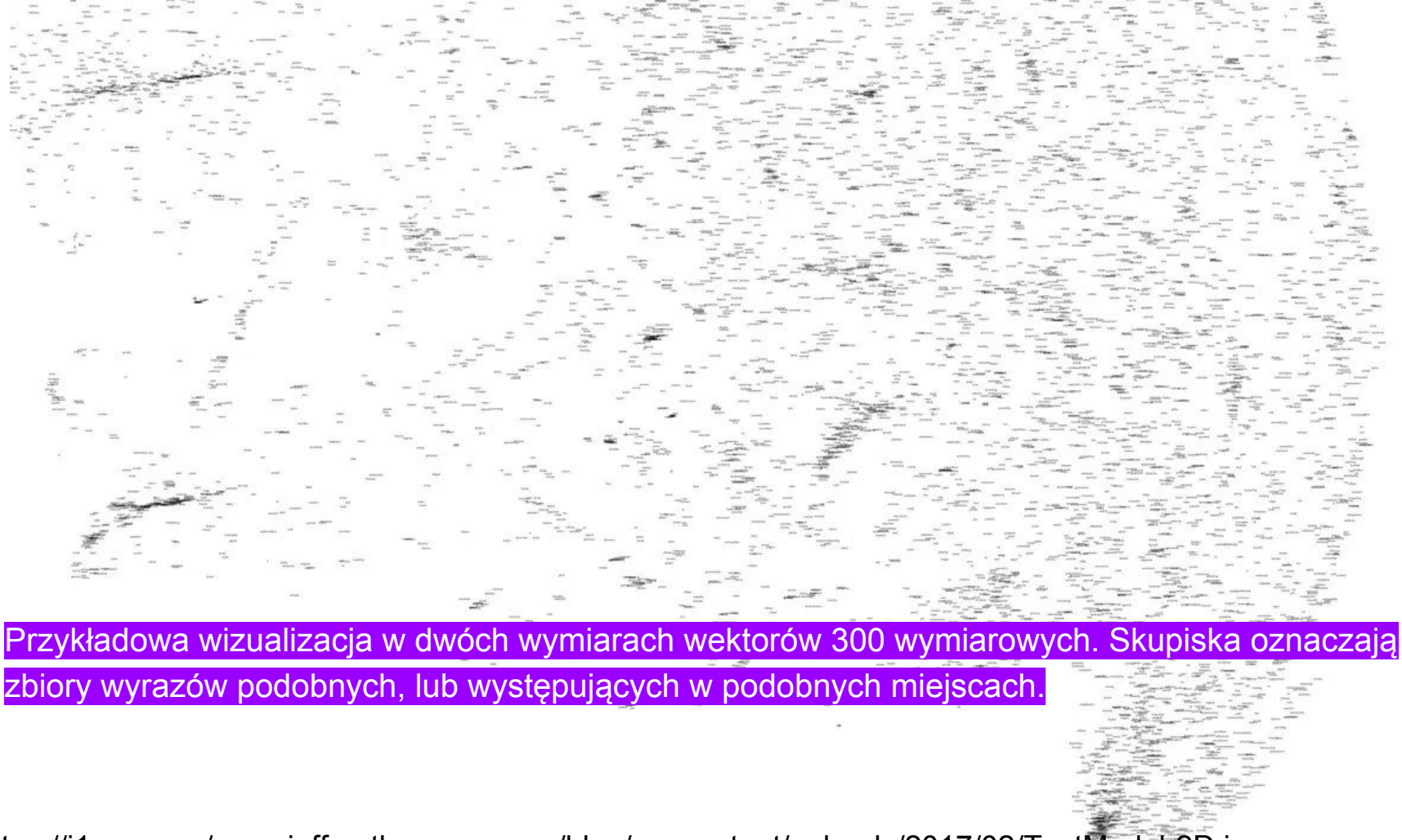
W przegladarce -> localhost:8888

Wizualizacja

Jak pokazac 300 wymiarowe wektory na kartce, ekranie?

Jest wiele metod redukcji wymiarow, zachowujacych strukture:

- PCA - Principal Component Analysis
- SNE - Stochastic Neighbour Embedding
- Sammon mapping
- CCA - Curvilinear Component Analysis
- ….



Przykładowa wizualizacja w dwóch wymiarach wektorów 300 wymiarowych. Skupiska oznaczają zbiory wyrazów podobnych, lub występujących w podobnych miejscach.

https://i1.wp.com/www.jeffreythompson.org/blog/wp-content/uploads/2017/02/TestModel-2D.jpg

Wizualizacja

- t-SNE = t-distributed stochastic neighbour embedding
- Algorytm nie jest nowy, ale konkretnie t-SNE opracowali G. Hinton i L.Maaten
- Intuicyjnie:
  - Skonstruuj rozklad prawdopodobienstwa miedzy parami obiektow wysoko-wy
    podobne maja duza szanse bycia wybranymi na “sasiada”, i na odwrot
  - Skonstruuj podobny rozklad prawdopodobienstwa, ale w 2 lub 3 wymiarach
  - Numerycznie minimalizuj dywergencje Kullbacka-Leiblera miedzy dwoma dys
    wzgledem lokalizacji na mapie 2d/3d

https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding

Demo:
2D t-SNE wygenerowane z TED talks + Word2Vec przy uzyciu biblioteki bokeh

https://github.com/Adamage/neural-nets-tutorials/blob/master/tensorflow-tutorials/natural-language/oxford-course-practical1/part4_tsne_of_ted.py

Skrypt należy uruchomić, powinienn sciągnąć dane, wytrenować word2vec a następnnie prostą mapę 2D t-sne, która jest interaktywna.

Outputs: webpage using **bokeh** library - interactive 2d map

Demo:
2D t-SNE wygenerowane z TED talks + Word2Vec, Przy uzyciu tensorflow i tensorboard

https://github.com/Adamage/neural-nets-tutorials/blob/master/tensorflow-tutorials/natural-language/word2vec/tensorboard_word2vec_TED.py

Wymaga scipy, numpy, tensorflow 1.2+

Outputs: tensorflow checkpoints i plik z metadanymi - nalezy uzyc tensorboard

Zrodla, tutoriale, ciekawe linki

Deep learning and machine learning:

- https://github.com/oxford-cs-deepnlp-2017/lectures
- http://cs224d.stanford.edu/syllabus.html
- https://www.quora.com/Deep-Learning-What-is-meant-by-a-distributed-representation
- http://www.cs.toronto.edu/~bonner/courses/2014s/csc321/lectures/lec5.pdf
-

Word vectors:

- https://www.tensorflow.org/tutorials/word2vec

T-SNE

- Publikacja autorow metody: http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf
- https://lvdmaaten.github.io/tsne/
- https://github.com/oreillymedia/t-SNE-tutorial
- http://alexanderfabisch.github.io/t-sne-in-scikit-learn.html
- http://distill.pub/2016/misread-tsne/