Do tej pory omówiliśmy BoW oraz sposób kodowania tego algorytmu słowo jest zakodowane przez wektor długości słownika zawierające jedną jedynkę dlatego BoW miało przechowywać indeksy na których ma pojawić się jedynka.

W zasadzie a co jakby udało się stworzyć taką reprezentację słów gdzie wartości byłyby wartościami rzeczywistymi ale wektor byłby krótszy a wartości muszą odzwierciedlać zależności między słowami- takie podejście nazywa się Word Embedding-> ma na celu wyznaczyć gęste wektory reprezentujące słowo. Takie techniki muszą oczywiście rzutować słowo na wektor liczb rzeczywistych, zmniejszać wymiarowość danych oraz uwzględniać zależności między słowami.

Najpopularniejsze jest podejście word2vec -> wprowadził w 2012 roku rewolucje w tej tematyce.

Word2Vec to model prosta sieć neuronowa warstwa wejściowa ukryta i wyjściowa. Wejścia i wyjścia są wielkości słowników a warstwa ukryta posiada liniową funkcję aktywacji a wyjście softmax. Stosuje się liniową funkcję aktywacji w warstwie ukrytej w teorii nie ma to sensu w pdf trochę wyjaśnień. Może to przypominać SVD, PCA i autoenkodery.

Tutaj będziemy chcieli stworzyć to w taki sposób aby podać słowo i otrzymać informacje o jego kontekście. Wyróżniamy dwa warianty C-Bow i Skip-gram.

C-Bow-> bierzemy kilka słów z otoczenia i na ich podstawie chcemy przewidzieć słowo które wybraliśmy.

Skip-gram-> wstawiamy słowo a sieć ma przewidzieć słowa które maja być w jego otoczeniu.

Tutaj trzeba pamiętać że BoW daje nam informację o całym dokumencie a tutaj tworzymy reprezentację dla pojedynczego słowa żeby to zrobić to na podstawie zdania utworzymy zbiór uczący.

Pierwsza rzecz to ustalenie otoczenia mamy zadanie

„Data mining is a dun for all of us”

Przyjmujemy że otoczenie wynosi dwa-> wybieram słowa i biorę po dwa słowa przed i po nim. Możemy sobie wyobrazić że na sieć będziemy podawać wektor dla słowa mining a oczekiwać wyjścia data.

Wprowadzamy wektor więc w tym przypadku wektor wejściowy będzie miał tyle jedynek ile słów się w otoczeniu znajduje

Podejście skip-gram jest przeciwne-> wybieramy jedno słowo i chcemy przewidzieć otoczenie.

Skip-gram jest wolniejszy podczas trenowania ale lepiej działa na małych zbiorach, C-BoW jest szybszy bo przetwarza wiele danych jednocześnie i działa lepiej na dużych zbiorach danych i może być mniej dokładny dla słów rzadko występujących.

Uczenie jest podobne co do klasycznych sieci-> funkcja straty najczęściej entropia krzyżowa, stosujemy jednak dwa triki przed wyznaczeniem błędu największą wartość softmax ustawiamy na 1 resztę na 0, stosujemy Negative sampling.

Negative sampling-> pomyślmy jak działa entropia krzyżowa mamy warstwę softmax, jest tylko jedne neuron który powinien zostać aktywowany czyli jedne przeciwko całemu słownikowi korzystniejsze byłoby wygaszenie neuronów niż znalezienie tego jednego. W przypadku negative sampling stosujemy modyfikacje tej metody i nazywa się to binarna entropia krzyżowa-> może być stosowana wszędzie gdzie do przewidzenia mamy jedną wartość.

Adaptujemy to do negative samplingu bierzemy jedną parę x y będziemy to nazywać przypadkiem pozytywnym i liczymy entropię krzyżową tylko dla jednego neuronu odpowiadającemu temu słowu y które powinno być aktywowane ale bierzemy również n neuronów które nie powinny znaleźć się w kontekście słowa. 1 pozytywny vs 20 negatywnych.

W efekcie otrzymujemy ukrytą reprezentację danych można je zwizualizować

Word2vec w tych sieciach nie chodzi o jak najlepsze przewidywanie słów a o wydobycie tej gęstej reprezentacji ze stanu ukrytego dla każdego ze słów co w przyszłości powinno zastąpić te długie wektory zakodowane w stylu on-hot encoding