Word Embeddings - gęste wektory reprezentujące znaczenie słów

mgr inż. Dawid Wisniewski - Przetwarzanie języka naturalnego

22 March 2021

Laboratoria 4 i

Wcześniejsze laboratoria: wektorowe reprezentacje słów, tokeny/n-gramy w kolumnach.

Niezła skuteczność w analizowanych przykładowych problemach klasyfikacji.

Laboratoria 4 ii

Na potrzeby kolejnych slajdów, weźmy jako przykład dwa dokumenty:

- Dokument1: Ala ma kota
- Dokument2: Alicja posiada kotka

Laboratoria 4 iii

Reprezentacje, które poznaliśmy nie zawierają informacji o znaczeniu słów, zliczają one jedynie ile razy dany ciąg znaków pojawił się w tekście i zapisują odpowiednią informację w wektorze liczb.

Jeśli posiadamy słownik utworzony z powyższych dokumentów: {Ala, ma, kota, Alicja, posiada, kotka}, to reprezetnacje bag-of-words tych dokumentów wyglądają następująco:

- Dokument1 [1, 1, 1, 0, 0, 0]
- Dokument2 [0, 0, 0, 1, 1, 1]

Nie da się z tych wektorów wywnioskować, że kotek, to coś podobnego do kota, Alicja to duża Ala, a ’ma’ i ’posiada’ to synonimy.

Laboratoria 4 iv

Jak zatem reprezentować słowa, aby zawrzec w tej reprezentacji informacje o znaczeniu?

Brytyjski lingwista John R. Firth w 1957 roku wypowiedział zdanie, które miało wielki wpływ na NLP:

On powiedział

You shall know a word by the company it keeps

Przykład

Kupiłem x8sadf23, ma dobrą kartę graficzną i idealnie mieści się do plecaka.

Z bardzo dużym prawdopodobieństwem, x8sadf23 to jakiś laptop.

Laboratoria 4 vi

Cwane, ale jak zrobić z tego wektory?

1. Weźmy bardzo duży zbiór dokumentów, im więcej tym lepiej.
2. Ustalmy sobie rozmiar okienka, np. 4 tokeny w lewo i 4 w prawo od aktualnego słowa (tokenu).
3. Przeczytajmy te dokumenty token po tokenie, sprawdzając z czym (z jakimi tokenami) dany token współwystępował wewnątrz okienka.

Laboratoria 4 vii

Zapiszmy informacje o współwystąpieniach za pomocą macierzy współwystąpień.

Otrzymamy bardzo dużą macierz. Ponieważ dowolne słowo, w teorii może współwystępować z dowolnym innym słowem, rozmiar tej macierzy to: **ilosc unikalnych tokenów we wszystkich dokumentach razem do kwadratu** (czyt. dużo więcej niż byśmy sobie tego życzyli).

Jeśli różnych tokenów we wszystkich dokumentach jest 20000, macierz ta, zaimplementowana naiwnie (nie jako sparse matrix), będzie prawdopodobnie rozmiaru 20000 * 20000 * ilość bajtów do przechowania potencjalnie dużych liczb (załóżmy 4 bajty), czyli około 1.5 GB.

Laboratoria 4 viii

Jednak wiersze (bądź kolumny przed podjęciem kolejnych kroków) będą miały 2 ważne cechy:

- Pojedynczy wektor będzie bardzo długi
- Ale będzie miał w sobie informację o znaczeniu, gdyż zawrze informacje o kontekście w jakim słowo było widziane w korpusie

kot i kotek będą współwystępować z podobnymi słowami, ich wektory będą miały wartości niezerowe w podobnych miejscach.

kot i kotek współwystępują:

- często: kuweta, pazury, karma
- rzadko: DiscoPolo, Wiktuał, Przemek

Laboratoria 4 ix

Używając odpowiedniej miary możemy zmierzyć to podobieństwo. Bardzo często używa się tzw. podobieństwa kosinusowego, która reprezentuje kosinus kąta pomiędzy dwoma wektorami.

$$similarity(\vec{v1}, \vec{v2}) = \frac{\vec{v1} \cdot \vec{v2}}{\|\vec{v1}\|\|\vec{v2}\|}$$

Innymi słowy: podobieństwo kosinusowe to ułamek; dzielimy sumę iloczynów odpowiadających sobie elementów wektora przez iloczyn długości (euklidesowych) tych wektorów.

Tam, gdzie dwa wektory na tej samej pozycji będą miały wartości niezerowe o tym samym znaku, tam będzie wyższa wartość miary kosinusowej. Im więcej takich sytuacji tym wyższe podobieństwo.

Redukcja wymiarowości.

Okazuje się, ze tak długie wektory, które otrzymaliśmy możemy ßkompresować", tj. zmniejszyc ich długość zachowując tylko najważniejsze informacje zakodowane w pierwotnym wektorze. Istnieje wiele sposobów na redukcję wymiarowości, dobrym źródłem infomracji jest np. (https://pbiecek.github.io/NaPrzelajDataMiningR/part-2.html lub http://www.cs.put.poznan.pl/jstefanowski/gi/redukcjawymiarowPCA.pdf).

Wtedy, znaczenie każdego słowa można reprezentować za pomocą zadanej liczby elementów w wektorze (np. 300), gdzie wektor 300-elementowy jest skompresowaną reprezentacją pierwotnego, wynikającego z macierzy współwystąpień.

Word embeddingi

Word embeddingi to reprezentacje wektorowe znaczeń słów, gdzie wektory są gęste (mają bardzo mało elementów zerowych) i o ograniczonej długości (najczęściej z zakresu 50 - 1000 elementów).

Okazuje się, że word embeddingi mają ciekawe własności, gdyż kodują nie tylko znaczenie jako takie, ale za pomocą dodawania i odejmowania wektorów, możemy wyznaczyć analogie (patrz jedno z zadań na dzisiejszy tydzień).

Laboratoria 4 xii

Podejścia oparte o sieci neuronowe.

W ostatniej dekadzie pojawiły się dużo wydajniejsze metody generowania embeddingów niż przedstawiona na wcześniejszych slajdach. O ile ogólna zasada działania pozostaje podobna, tak z użyciem sieci neuronowych udało się zmniejszyć wymagania pamięciowe i czas na generowanie dobrych reprezentacji słów. Nadal jednak wcześniej na slajdach przedstawiona historia jest najprostszym wprowadzeniem do idei word embeddingów.

Najbardziej popularne metody wyznaczania embeddingów to GloVe i Word2Vec (https://www.cs.put.poznan.pl/alawrynowicz/PJN_4-2019.pdf, https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf)

Laboratoria 4 xiii

W internecie znajdziemy zbiory już wygenerowanych embeddingów, np. na stronie: https://nlp.stanford.edu/projects/glove/ znajdziemy różnej długości wektory stworzone za pomocą GloVe.

Laboratoria 4 xiv

Ale chwila, mamy reprezentacje pojedynczych słów, jak to wykorzystać do klasyfikacji dokumentów, gdzie każdy dokument powinien być wektorem o równej długości?

Mając sekwencję słów(tokenów) w dokumencie, gdzie dla każdego słowa(tokenu) posiadamy embedding, zawsze tych samych rozmiarów, (np. wektor 300 liczb), możemy cały dokument reprezentować jako środek ciężkości wszystkich tych embeddingów:

Laboratoria 4 xv

1. Dla danego dokumentu wykonaj tokenizację, zbierz embeddingi przypisane poszczególnym tokenom (np. z pretrenowanego zbioru embeddingów z https://nlp.stanford.edu/projects/glove/) iterując token po tokenie w dokumencie
2. Mając zbiór tokenów i przypisanych do nich embeddingów (każdy o zadanej długości, np 300), wyznacz wektor o długości 300, reprezentujący cały dokument, gdzie:
   - 2.1 na 1 pozycji wektora zapisujemy średnią arytmetyczną wszystkich pierwszych pozycji embeddingów dla tokenów z dokumentu
   - 2.2 na 2 pozycji wektora zapisujemy średnią arytmetyczną wszystkich drugich pozycji embeddingów dla tokenów z dokumentu
   - 2.3 ...i tak dalej, aż do 300 elementu

Laboratoria 4 xvi

Okazuje się, że tak prosta metoda, gdzie znaczenie dokumentu to ’średnie’ znaczenie zawartych w nich słów, daje często bardzo dobre rezultaty! Oczywiście nie jest to jedyna metoda (dobry przegląd znajdziecie pod adresem: https://towardsdatascience.com/document-embedding-techniques-fed3e7a6a25d), ale idealna w swojej prostocie do wprowadzenia w problematykę i często wykorzystywana w praktyce z uwagi na prostotę implementacji.

(BONUS): **Odległość Levenshteina**: na naszych laboratoriach wykorzystywać będziemy też odległość Levenshteina do automatycznego poprawiania literówek w tekście. O ile sama metoda wyznaczania odległości edycyjnej (często używana alternatywna nazwa dla odległości Levenshteina) jest już zaimplementowana w wielu narzędziach, o tyle warto zapoznać się z jej działaniem. Pod adresem: https://pl.wikipedia.org/wiki/Odleg%C5%82o%C5%9B%C4%87_Levenshteina znajdziecie Państwo przystępny opis.