mikołaj morawiec

WIEiT AGH
Informatyka III r.
Pracownia Projektowa

Analiza sentymentu recenzji przy użyciu rekursywnych sieci neuronowych

DOKUMENT PROJEKTOWY

1 Wstęp

1.1 Opis koncepcji

Idea stojąca za implementowanym w projekcie modelem do analizy sentymentu zdań jest oparta na pracy „Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks" (R. Socher *et al.*) [1] i polega na wykorzystaniu rekursywnej sieci neuronowej opartej na drzewie parsingu analizowanego zdania, zamiast bardziej klasycznego modelu z sekwencyjnym podawaniem kolejnych wyrazów do bramki. Wspomniana praca zajmowała się sieciami z wykorzystaniem bramki LSTM. Celem tego projektu jest realizacja tej samej idei z prostszymi bramkami GRU, co powinno zapewnić lepszą wydajność w stosunku do LSTM.

1.2 Założenia

Sieć działa na zdaniach w języku angielskim. Za pojedyncze słowo uznaje się wszystko co znajduje się pomiędzy dwiema spacjami. Dodatkowo, znaki interpunkcyjne oraz nawiasy również uznaje się za pojedyncze słowa, posiadające swój sentyment.

Do uczenia i testowania sieci służy zbiór Stanford Sentiment Treebank (ozn. SST) [3], podobnie jak w pracy [1]. Pozwala to na porównanie uzyskanej skuteczności z wynikami uzyskanymi dla bramki LSTM.

Dla praktycznych zastosowań wyuczonego modelu stosuje się zewnętrzny parser języka naturalnego (Stanford Parser, [5]).

2 Model

2.1 Ogólny opis

Podstawowym elementem składowym sieci neuronowej w zastosowanym modelu jest bramka GRU. Zastosowano wersję bramki specjalnie dostosowaną do pracy w strukturze binarnego drzewa parsingu, tj. bramkę pobierającą dwa wektory stanu ukrytego (od dzieci w drzewie) i przekazującą jeden taki wektor do rodzica w drzewie.

Każda z bramek jest również zaopatrzona w moduł zamieniający stan ukryty na wektor prawdopodobieństw możliwy do porównania z sentymentem pobranym ze zbioru SST. Moduł zasadniczo opiera się na pojedynczej warstwie sieci redukującej wymiar wektora oraz funkcji transferu do prawdopodobieństw.

W liściach stosowana jest dużo prostsza bramka nazwana bramką wejściową. Jest ona również zaopatrzona we wspomniany moduł z funkcją transferu, co pozwala na uzyskanie wyników dla pojedynczych słów.

2.2 Reprezentacja wektorowa słów

Pojedyncze słowa podawane są na wejście w sieci w postaci wektorów. Do przekształcenia słów języka naturalnego na rzeczone wektory wykorzystuje się słownik przyporządkowujący każdemu ze znajdujących się w nim słów inny wektor.

Wykorzystano reprezentację wektorową słów pochodzącą z Global Vectors for Word Representation (GloVe) [7]. W testach korzystano z wektorów o wymiarowości $N = 300$. Słownik posiada ok. 2 mln słów, co wystarcza dla potrzeb praktycznych. Nie wszystkie słowa mają swoją reprezentację w słowniku, dla brakujących słów ustala się zupełnie losową reprezentację.

2.3 Bramka Wejściowa

Do przetworzenia reprezentacji wektorowej słów w liściach drzewa projekt wykorzystuje zwykłą sieć jednowarstwową. Sieć przetwarza wektor wejściowy o x o wymiarze N na wektor o mniejszej

wymiarowości, odpowiadającej wymiarowości stanu ukrytego w bramce GRU (ozn. M).

$$h_i = \tanh(W_1^x x_i) \tag{2.3.1}$$

Indeksem i oznaczono indeks przypisany do kolejnych słów w zdaniu.

2.4 Bramka GRU

W projekcie wykorzystano bramkę binarną bramkę GRU, pobierającą stany ukryte od dwóch dzieci w drzewie. Równania oparto na pracach [6], [1] oraz [2]. W bramce jest tworzony stan-kandydat zgodnie z następującym zestawem równań:

$$\begin{aligned} r &= \sigma(W_1^r h_{1,t-1} + W_2^r h_{2,t-1}) \\ \bar{h} &= \tanh(W_1^h(h_{1,t-1} \circ r) + W_2^h(h_{2,t-1} \circ r)) \end{aligned} \tag{2.4.1}$$

Stany ukryte dzieci h_t w drzewie parsingu oznaczono symbolami $h_{1,t-1}$ i $h_{2,t-1}$. W tym przypadku indeks t oznacza stopień zagłębienia w drzewie.

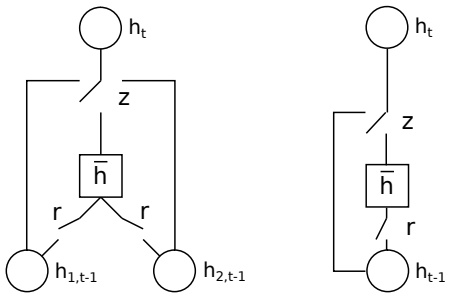
Następnie bramka rozważa przyjęcie stanu-kandydata jako stanu wyjściowego, wraz z oboma stanami wejściowymi. Za proporcje wkładów tych trzech opcji odpowiada bramka z:

$$z^i = \sigma(W_1^{z,i} h_{1,t-1} + W_2^{z,i} h_{2,t-1}) \tag{2.4.2}$$

W praktyce bramka z jest implementowana jako zestaw dwóch bramek. Ostatecznie otrzymuje się stan wyjściowy:

$$h_t = (1 - \sum_i z^i) \circ \bar{h} + z^i \circ h_{i,t-1} \tag{2.4.3}$$

Do jego zamiany na tensor prawdopodobieństw (ang.: *log-likelihood*) wykorzystuje się funkcję softmax, wraz z pojedynczą warstwą sieci do redukcji wymiaru z M do C, gdzie C oznacza liczbę klas (bramka wyjściowa).



Rysunek 1: Ilustracja bramki GRU (po lewej), porównana z bramką GRU sekwencyjną według [6]

2.5 Bramka wyjściowa

2.5.1 Funkcje transferu

Podstawową funkcją transferu w projekcie była funkcja softmax, połączona z pojedynczą warstwą dla redukcji wymiaru stanu ukrytego M do liczby klas C.

2.5.2 Dropout

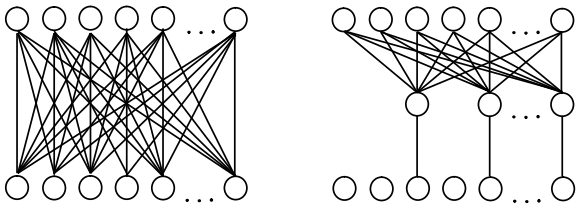
Testy wykazały że zastosowanie warstwy dropout, tj. funkcji maskującej część wejścia dla wa w module funkcji transferu pozwala na uzyskanie nieco lepszych wyników, rzędu kilku punktów procentowych skuteczności. Zastosowany w projekcie moduł usuwa połowę składowych wejścia i mnoży

pozostałe przez 2. Usunięcie polega na zastąpieniu składowej wartością 0. Składowe do usunięcia są losowane (schemat Bernoulliego) spośród wszystkich składowych wektora wejściowego.

Można zapisać wyjście y_t z bramki GRU o stanie ukrytym h_t za pomocą następujących równań:

$$\begin{aligned} h'_t &= \text{dropout}(h_t) \\ y_t &= \log(\text{softmax}(W_1^y h'_t) \end{aligned} \tag{2.5.1}$$

Działanie dropout ilustruje rysunek 2. Po lewej przedstawiono warstwę sieci bez dropout, po prawej usunięto część wejścia.



Rysunek 2: Ilustracja działania dropout

2.6 Funkcja kosztu NLLC

Do uczenia wykorzystuje się kryterium NLLC (*negative log-likelihood*), odpowiednie do problemu klasyfikacyjnego. Tensor prawdopodobieństw uzyskuje się ze stanu ukrytego bramki GRU wstawia do kryterium, wraz indeksem odpowiadającym poprawnej klasie do której powinien być zakwalifikowany dany przypadek.

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n} p_i + \frac{\lambda}{2}||\theta||^2 \tag{2.6.1}$$

Parametr λ mnożony przez kwadrat normy wektora parametrów służy do regularyzacji kryterium. Suma przebiega po wszystkich węzłach drzewa i jest obliczana w rekurencyjny sposób, podczas propagacji do przodu. Symbolem p^i oznaczono prawdopodobieństwa uzyskane z funkcji softmax (rów. wyżej) uzyskane dla poprawnej klasyfikacji:

$$p_i = y_i^c \tag{2.6.2}$$

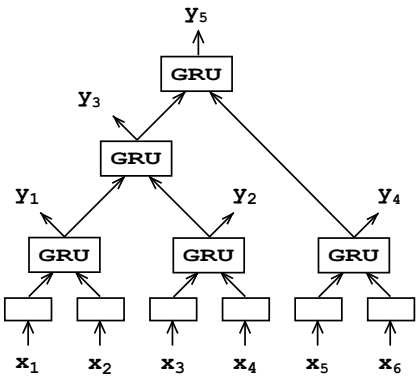
gdzie $c \in \{1, ...C\}$ oznacza poprawną klasyfikację danej frazy lub słowa pobraną z SST. Sposób ustalania c podano niżej, w sekcji dotyczącej uczenia sieci (rów 5.2.1).

3 Struktura drzewa

Strukturę drzewa osiągnięto za pomocą odpowiedniego mechanizmu obliczającego wejścia dla węzłów w drzewie po kolei, zaczynając od liści, tj. pojedynczych słów. Wejścia podawane są do sieci w kolejności postorder DFS.

Uczenie odbywa się algorytmem BTS. Dla każdego węzła drzewa tworzona jest kopia sieci neuronowej. Dla każdego węzła wykonywana jest propagacja do przodu i wstecz, wszystko odbywa się kolejności wynikającej z przebiegu algorytmu DFS na drzewie, przy czym dzieci po lewej stronie (wczejśiej w zdaniu) eksplorowane są jako pierwsze. Algorytm BTS jest wykonywany rzazem na całym drzewie, tj. uczenie odbywa się zarówno na parametrach bramki GRU, jak i na parametrach bramki wejściowej.

Obrazek obok ilustruje strukturę pełnej sieci. Indeksy oznaczają kolejność obliczania kolejnych wyjść przy propagacji do przodu.



Rysunek 3: Drzewiasta sieć neuronowa

4 Parametry modelu

Podsumowując dyskusję modelu można stwierdzić, że model posiada następujący zestaw podstawowych parametrów:

C	liczba klas
N	wymiar przestrzeni reprezentacji wektorowej słów języka naturalnego
M	wymiar przestrzeni wektorowej stanów ukrytych bramki GRU
ϵ	stała uczenia
λ	stała regularyzacji

Zasadniczo nie korzystano z reprezentacji o N innym niż 300, więc pierwszy parametr jest właściwie nieistotny. Parametr C jest elementem definicji problemu klasyfikacyjnego (wynosi 2 lub 5). Pozostałe trzy wpływają na skuteczność konkretnej instancji modelu, i pozwalają porównywać skuteczności w obrębie ustalonego N i C.

5 Uczenie, Selekcja najlepszej sieci

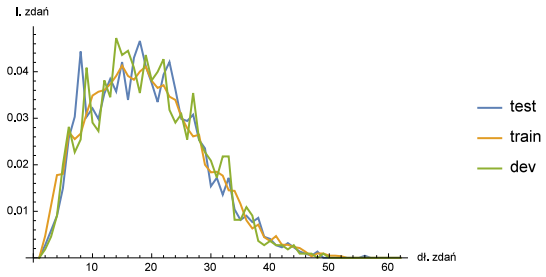
5.1 Charakterystyka SST

Uczenie odbywa się na wspomnianym zbiorze SST. Zbiór jest podzielony na trzy części: treningową, optymalizacyjną i testową. Cały zbiór zawiera łącznie 11855 zdań zbudowanych z 14926 różnych wyrazów. Należy zaznaczyć że jako wyrazy traktowane są również znaki interpunkcyjne, nawiasy itd. Najkrótsze ze zdań ma dwa wyrazy, najdłuższe zdanie ma 56 wyrazów co przekłada się na drzewo parsingu o 111 wierzchołkach. Zdania w zbiorze pochodzą w większości z różnorodnych recenzji tekstów kultury.

Reprezentacja wektorowa słów jest pobierana z pliku GloVe. Pojawiają się braki w słowniku, dotyczące przede wszystkim nazw własnych i nazwisk. Poza tym słownik nie zawiera reprezentacji słów złożonych z pauzami jak na przykład „prep-school", „not-very-funny", „skyscraper-trapeze". Jak widać po przykładzie, niektóre z tych słów niosą ze sobą sentyment, więc ich brak w słowniku może być znaczący.

Zbiór treningowy jest zawiera 8544 zdań. Służy do uczenia sieci zgodnie z opisanym wyżej algorytmem BTS. Zbiory optymalizacyjny i testowy są wykorzystywane tylko do propagacji w przód (ewaluacji). Zbiór optymalizacyjny zawiera 1101 zdań i jest wykorzystywany do oceny skuteczności sieci w trakcie procesu uczenia (po każdej z epok). Zbiór testowy zawiera 2210 zdań. Służy wyłącznie do ostatecznej oceny skuteczności wyuczonej sieci.

Upewniono się że pod podział SST jest poprawny ze względu na długość zdań w nim zawartych (rys 4).



Rysunek 4: Histogramy długości słów dla poszczególnych zbiorów testowych, znormalizowane do jedynki

5.2 Wartości sentymentu

SST zawiera oprócz zdań wartości sentymentu przypisane do każdej z fraz jak i do całego zdania. Należy nadmienić, że zdania posiadają gotowe drzewa parsingu zapisane w zbiorze w notacji parent-pointer (pol.: notacja spaghetti). Sentyment jest przedstawiony jako liczba z zakresu $(0, 1]$.

Funkcja kosztu jest obliczana na podstawie prawidłowych wartości sentymentu, przy czym konieczna jest ich konwersja z wersji ciągłej do dyskretnej, realizowana odpowiednią funkcją, w zależności od rozpatrywanego wariantu problemu (tj liczby klas):

$$f_V(x) = \begin{cases} 1 & x \in (0, 0.2) \\ 2 & x \in [0.2, 0.4) \\ 3 & x \in [0.4, 0.6) \\ 4 & x \in [0.6, 0.8) \\ 5 & x \in [0.8, 1] \end{cases} \quad \text{albo} \quad f_{II}(x) = \begin{cases} 1 & x \in (0, 0.4] \\ 2 & x \in (0.6, 1] \end{cases} \tag{5.2.1}$$

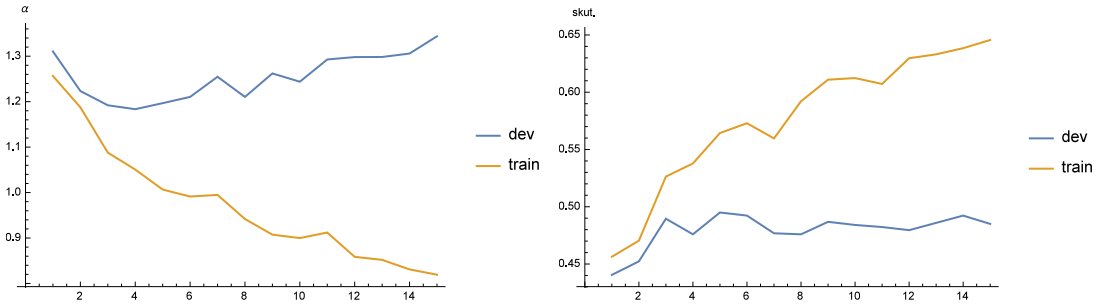
Tę samą operację stosuje się przy obliczaniu skuteczności przy ewaluacji.

5.3 Proces uczenia

Uczenie odbywa się na podzbiorze treningowym SST. Kolejność podawania zdań do sieci jest ustalana od nowa w losowy sposób dla każdej kolejnej epoki, efektem czego jest fakt, iż procesy uczenia nie są powtarzalne dla ustalonego zestawu parametrów sieci. Ten fakt wskazuje na stochastyczny aspekt całego procesu uczenia i testowania.

Do selekcji optymalnie wyuczonej sieci stosuje się przeznaczoną do tego celu część optymalizacyjną SST. Wybiera się sieć po epoce w której osiągnęła ona najlepszy wynik (tj. skuteczność dla całych zdań).

W testach stosowano przeważnie 25 epok, wybierając sieć o najlepszym wyniku w działaniu na wspomnianej części zbioru. Proces selekcji ilustruje wykres:



Rysunek 5: ilustracja 15 epok uczenia. Po lewej stronie koszt, po prawej stronie skuteczność sieci na całych zdaniach

W powyższym przypadku sieć po epoce nr 5 zostałaby wybrana jako wynik procesu uczenia i przekazana do testowania. Na wykresach zilustrowano również koszt i skuteczność sieci na zbiorze treningowym.

Literatura

[1] „Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks", Kai Sheng Tai, Richard Socher, Christopher D. Manning, (Stanford University 2015)

[2] „On the Properties of Neural Machine Translation: Encoder–Decoder Approaches", Kyunghyun Cho Bart van Merriënboer, Dzmitry Bahdanau, (Université de Montréal 2014)

[3] Stanford Sentiment Treebank: `http://nlp.stanford.edu/sentiment/index.html`

[4] „Empirical evaluation of gated recurrent neural networks on sequence modeling", Chung, Junyoung, *et al.* (2014)

[5] Stanford Parser: `http://nlp.stanford.edu/software/lex-parser.shtml`

[6] „Neural machine translation by jointly learning to align and translate", D. Bahdanau *et al.*

[7] Global Vectors for Word Representation: `http://nlp.stanford.edu/projects/glove`