1 WBST

Testy przeprowadziłem używając jednego z udostępnionych testów WBST dla słów pojajwiających się przynajmniej 1000 razy. Problemem było to, że wszystkie zbiory do testów były w języku polskim, podczas gdy moje wszystkie modele były uczone na korpusach w języku angielskim. Rozwiązałem ten problem przetłumaczając zbiór z języka polskiego na angielski. Każdy model zamienia słowo na 100-wymiarowy wektor. Podobieństwo wektorów wyznaczałem za pomocą podobieństwa kosinusowego. W tabeli przedstawiam tylko miarę dokładności, ponieważ miary dokładności, precyzji, recall i f1 dla każdego modelu były prawie takie same.

model	accuracy
`full_m1`	0.576
`full_m2`	0.595
`sample_m1`	0.542
`sample_m2`	0.546

Jak widać lepsze wyniki daje model uczony na korpusie ogólnym, ponieważ zakres jego słownictwa bardziej przypomina zakres słownictwa korpusu plWNC (po przetłumaczeniu) na podstawie którego stworzony został test. W tym teście także lepiej poradziły sobie modele skipgram (m2) w porównaniu do cbow.

2 Klasyfikacja

Do eksperymentów użyłem 3 klasyfikatory oparte na sieciach neuronowych. Pierwszy z nich to konwolucyjna sieć neuronowa. Taka sieć neuronowa wymaga danych wejściowych o stałym kształcie, co wymusza padowanie lub przycinanie każdej wypowiedzi, żeby zrównać długość. Każdą wypowiedź zrównywałem do 100 tokenów. Nadmiarowe tokeny obcinałem z końca, a brakujące padowałem wyzerowanymi wektorami.
Architektura sieci CNN:

```
nn.Conv1d(100, 100, 5),
nn.ReLU(),
nn.MaxPool1d(kernel_size=2, stride=2),
nn.Conv1d(100, 100, 3),
nn.ReLU(),
nn.MaxPool1d(kernel_size=2, stride=2),
nn.Conv1d(100, 100, 2),
nn.ReLU(),
nn.MaxPool1d(kernel_size=2, stride=2),
nn.Flatten(),
nn.Linear(1100, 2)
```

Druga sieć to sieć RNN, podobna do tej wykorzystywanej w 2. zadaniu, ale przystosowana do klasyfikacji. Jest ona zbudowana z 3 warst RNN przedzielonych warstwami ReLU i jednej warstwy liniowej na końcu do klasyfikacji. Zaletą tej sieci jest to, że może ona jako dane wejściowe przyjąć sekwencję o dowolnej długości, dzięki czemu w tym wypadku nie było

konieczne przycinanie, ani padowanie sekwencji. Z drugiej strony zmodyfikowana metoda ładowania danych zmiennej długości powodowała znaczne spowolnienie pętli uczącej (nawet do 50 razy w porównaniu z CNN).
Architektura sieci RNN:

```
nn.RNN(100, 100, 3, batch_first=True),
nn.Linear(100, 2)
```

Ostatnim klasyfikatorem jest sieć LSTM. Działa on podobnie do klasyfikatora RNN, ale warstwy RNN zostały zastąpione warstwami LSTM. Korzysta też z tego samego sposobu ładowania danych jak RNN. Architektura sieci LSTM:

```
nn.LSTM(100, 100, 3, batch_first=True),
nn.Linear(100, 2)
```

Ze względu na długi czas uczenia, wszystkie sieci były uczone przez 10 epok. Proporcje zbiorów treningowego do walidacyjnego to 4:1. Poniższe wyniki są zaprezentowane dla epok o najlepszej metryce f1 na zbiorze walidacyjnym.

model	klasyfikator	precyzja	recall	f1
`full_m1`	RNN	**0.518**	0.139	0.219
`full_m1`	LSTM	0.517	**0.172**	**0.256**
`full_m1`	CNN	0.487	0.169	0.251
`full_m2`	RNN	0.452	**0.867**	**0.594**
`full_m2`	LSTM	**0.523**	0.228	0.318
`full_m2`	CNN	0.520	0.197	0.286
`sample_m1`	RNN	0.516	**0.177**	**0.263**
`sample_m1`	LSTM	**0.553**	0.138	0.221
`sample_m1`	CNN	0.517	0.143	0.224
`sample_m2`	RNN	0.493	0.138	0.215
`sample_m2`	LSTM	**0.522**	**0.221**	**0.310**
`sample_m2`	CNN	0.511	0.208	0.295

Jak widać o ile wyniki CNN nie były znacznie gorsze od pozostałych, to w każdej kategorii najlepiej radziły sobie klasyfikatory RNN i LSTM. Można to przypisać obsłudze przez te sieci sekwencji o zmiennej długości. Ta sama zmienna długość wymusiła modyfikacje data loadera, co znacznie je spowolniło. Zużycie pamięci, o ile nie było dokładnie mierzone, to obciążenie pamięci RAM i VRAM było podobne dla wszystkich sieci.
Wyniki RNN dla `full_m2` są wyraźnym outlierem. Jest to oznaka zjawiska, którego inaczej nie widać biorąc tylko najlepsze epoki. Jest nim niestabilność uczenia. Powoduje ona, że wyniki potrafią oscylować o nawet 0.1 między epokami. Szczególnie widoczne było to dla LSTM, gdzie pojawiały się epoki, gdzie całemu zestawowi danych klasyfikator przypisał klasę 0.