NLP - Dokumentacja końcowa

Bartłomiej Olber, Aleksander Samek

Temat projektu

Pretrenowanie modelu (XLNet) przystosowanego do określania interpretowalnego podobieństwa semantycznego (iSTS - https://alt.qcri.org/semeval2016/task2/) dwóch zdań w języku angielskim. Odniesienie się do wyników z projektu [3].

Definicja problemu

iSTS

Zadanie iSTS [1] (ang. interpretable semantic textual similarity) ma na celu określenie stopnia bliskości znaczeniowej dla pary fragmentów (ang. chunks) zdań w języku angielskim, poprzez przyporządkowanie typu dopasowania pomiędzy fragmentami zdań oraz, równolegle, numerycznej oceny podobieństwa w skali 0-5.

Stopień dopasowania

Stopień podobieństwa dwóch fragmentów zdań jest określony na dwa sposoby:

Ocena numeryczna

- 0 - fragmenty zdań są zupełnie niezwiązane znaczeniowo ze sobą.
- 1,2 - fragmenty są "nieco" związane ze sobą.
- 3,4 - fragmenty mają podobne znaczenie.
- 5 - znaczenie fragmentów jest identyczne.

Typ podobieństwa

- EQUI - oba fragmenty mają to samo znaczenie.
- OPPO - znaczenia fragmentów są sobie przeciwstawne.
- SPE1 - oba fragmenty mają podobne znaczenie, ale fragment pierwszy jest bardziej szczegółowy niż fragment drugi.

- SPE2 - odwrotnie do SPE1.
- SIMI - oba fragmenty mają podobne znaczenie, ale nie można ich przydzielić do żadnej z powyższych relacji (EQUI, OPPO, SPE1, czy SPE2).
- REL - oba fragmenty są powiązane znaczeniowo, ale nie można ich przydzielić do żadnej z powyższych relacji.
- NOALI - oba fragmenty nie są ze sobą powiązane

W dokumentacji oraz kodzie [3] pojawia się jeszcze typ ALIC. Jest to artefakt z poprzedniej edycji zadania iSTS (SemEval 2015), który został w roku 2016 usunięty i nie występuje już w zbiorach danych, więc go nie opisujemy. Musimy jednak pozostawić go w kodzie, żeby otrzymać poprawne miary **F type**, **F score+type** podczas ewaluacji skryptami konkursowymi *evalF1_penalty.pl*, *evalF1_no_penalty.pl*.

Ocena a typ

Zbiory danych iSTS zostały oznaczone przez grupę ludzi, którzy arbitralnie przyznawali oceny numeryczne oraz kategoryczne stopniu podobieństwa semantycznego fragmentom zdań. Jedyne reguły dotyczące powiązania oceny numerycznej i typu są następujące:

- ocena podobieństwa powinna wynosić 0 wtedy i tylko wtedy, gdy przyporządkowany typ to NOALI,
- ocena podobieństwa powinna wynosić 5 wtedy i tylko wtedy, gdy przyporządkowany typ to EQUI

Typy SPE1 i SPE2 zostają przypisane do wartości 3 lub 4 w ocenie numerycznej, co sprawia że tracimy informację dotyczącą szczegółowości jednego zdania względem drugiego (tj. SPE1=SPE2).

```
[12] <=> [10] : (SIMILAR 4)
[killed] <=> [killed] : (EQUIVALENT 5)
[in bus accident] <=> [in road accident] : (MORE-SPECIFIC 4)
[in Pakistan] <=> [in NW Pakistan] : (MORE-GENERAL 4)
```

Studia literaturowe

XLNet [4] jest to sieć typu transformer oparta o architekturę Transformer-XL. Główny wkład autorów XLNet'u polega na zaproponowaniu efektywnego sposobu uczenia (pretrainingu) modelu językowego. Autoregresyjna technika uczenia (czyli taka, która ma na celu zrozumienie relacji pomiędzy elementem zdania tj. tokenem, a elementami go poprzedzającymi tj. kontekstem) opiera się na przetwarzaniu permutacji tekstu. Dzięki temu model jest w stanie estymować prawdopodobieństwo warunkowe wystąpienia danego elementu zdania w kontekście poprzedzających i następujących po nim innych tokenów. Koniecznym do poniesienia kosztem autoregresywnego uczenia permutacjami jest znacze wydłużenie czasu treningu. Model przetwarza daną sekwencję wielokrotnie - tyle razy, ile jest wygenerowanych permutacji.

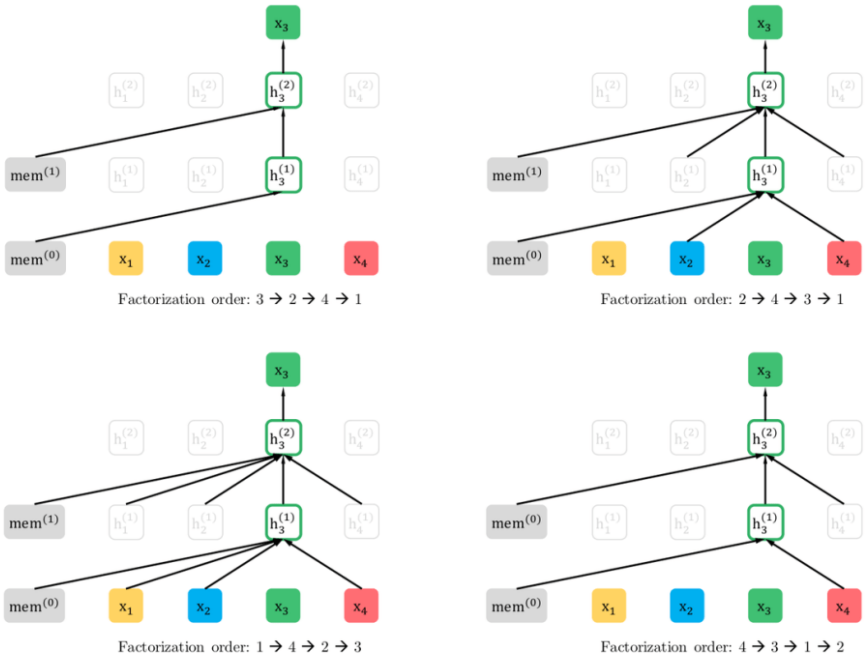


Figure 1: Illustration of the permutation language modeling objective for predicting x_3 given the same input sequence x but with different factorization orders.

Rysunek z artykułu [4] ilustrujący fakt, że model uczy się jakie jest prawdopodobieństwo warunkowe wystąpienia danego tokenu (X_3) w różnych kontekstach. Odpowiednio na rysunku: bez kontekstu; w otoczeniu tokenów X_2 i X_4; w otoczeniu tokenów X_1, X_2 i X_4; w otoczeniu tokenów X_4

Zalety wobec BERT [2]

Uczenie BERT'a polega na odtwarzaniu prawdziwych danych ze wybrakowanych. Czyli niektóre tokeny zastępowane są symbolem [MASK]. A transformer uczony jest uzupełniać owe braki. Podejście te posiada liczne zalety i sprawiło, że BERT przez pewien czas był "state-of-the-art", lecz posiada również dwie wady, na które, wg autorów XLNet'u, odpowiada XLNet.

1. Do konkretnych zastosowań bierze się pretrenowany model BERT, uczony na danych zawierających symbol [MASK] i dotrenowuje się go na danych "prawdziwych", stosownych do konkretnego zastosowania. Pretrenowany w ten sposób model ma tendencję do przekopiowywania niezamaskowanych tokenów i uzupełniania jedynie zamaskowanych. Jest to przydatna umiejętność w odtwarzaniu tekstu, ale nieco niepożądana przy różnorakich transformacjach tekstu.

2. Zamaskowane symbole są wyznaczane niezależnie od siebie. To znaczy, że podczas uczenia model nie bierze pod uwagę zależności pomiędzy zamaskowanymi tokenami. Jeśli w zdaniu są dwa symbole [MASK] maskujące elementy zdania silnie ze sobą powiązane, BERT uzupełniając równolegle pierwszą i drugą maskę do obu uzupełnień bierze jedynie pod uwagę jedynie niezamaskowane fragmenty sekwencji.

Opis rozwiązania

Koledzy Szaknis, Kulus i Strykowski [3] przygotowali notatnik Google Colab (.ipynb) z implementacją XLNetu dostosowaną do zadania iSTS. Rozszerzyli oni model z biblioteki HuggingFace o dwie głowy - regresji i klasyfikacji. Do głów dostarczany jest wektor aktywacji ostatniej warstwy ukrytej XLNet, a nastepnie obie głowy niezależnie przewidują numeryczną ocenę podobieństwa fragmentów zdań oraz klasyfikują typ podobieństwa - zgodnie z specyfikacją zadania iSTS. Przygotowali także kod do uczenia oraz ewaluacji modelu na danych iSTS.

Naszym zadaniem jest dodanie kroku pretrenowania na większym zbiorze danych, zawierającym parafrazy zdań, przed właściwym dotrenowaniem na danych iSTS. Zbiory zawierające parafrazy zdań są dostosowane do określania podobieństwa semantycznego pomiędzy całymi zdaniami (STS, semantic textual similarity). Spośród podanych przez Prowadzącego zbiorów wybraliśmy Quora Question Pairs zawierajacy ponad 400 tys. par pytań oznaczonych binarnie (1 - to samo znaczenie, 0 - różne znaczenie). Pretrenowanie mamy wykonać w dwóch konfiguracjach:

1. Pretrenowanie nienadzorowane, czyli autoregresyjny trening transformera na permutacjach zdań zgodnie z opisem z artykułu [4]. W tym wypadku nie używamy dwóch głów z rozwiązania [3]. Do fine-tuningu dla danych iSTS pretrenowany na QQP, XLNet zostanie wczytany, a neurony głów będą zainicjalizowane losowo.

2. Pretrenowanie dostosowane do zadania określania podobieństwa semantycznego. W tym wypadku zostanie użyty model z głowami, lecz należy zmienić wielkość głowy klasyfikacji (w iSTS mamy 7 klas, a w QQP 2). Po pretrenowaniu na QQP, analogicznie do pretrenowania nienadzorowanego, wagi właściwego XLNetu zostaną użyte do iSTS, ale warstwy głów muszą być zainicjalizowane losowo.

Implementacja

Podążając za pracą kolegów [3], dopisaliśmy kod do notatnika uwzględniający nowy zbiór danych. Językiem implementacji jest Python. Główne dwie biblioteki to HuggingFace, zawierająca XLNet oraz zbiór danych QQP oraz PytorchLightning, w której koledzy przygotowali silnik uczenia i ewaluacji iSTS. Nasz projekt znajduje się pod adresem (https://github.com/BartlomiejOlber/xlnet˙ists).

Bardzo ważnym elementem tego projektu jest dostęp do zasobów obliczeniowych. Pretrenowanie bardzo dużego modelu, jakim jest XLNet, na dużym zbiorze danych, jakim jest QQP, jest niewykonalne na Google Colab. Na szczęście posiadamy prywatną maszynę z dobrym GPU, dzięki czemu byliśmy w stanie wykonać kilka treningów.

Do wizualizacji uczenia się modelu wykorzystujemy platformę wandb.ai, a do obliczenia wyników F1 skrytpy perlowe przygotowane przez poprzedni zespół.

Instrukcja obsługi

Do przeprowadzenia pretrenowania i fine-tuningu należy uruchomić notatnik o nazwie *NLP_Proj.ipynb*. Przed włączeniem kodu należy wypełnić wszystkie pola pt. "ENTER_YOUR_VALUE". Są to zmienne związane z dostępem do logów uczenia ze strony Weights&Biases. W przypadku uruchamiania projektu ze środowiska Google Colab nie ma żadnych dodatkowych wymogów co do jego przygotowywania. W innych przypadkach należy zakomentować/usunąć dwie pierwsze komórki notatnika i zainstalować wszystkie biblioteki wymienione w pliku *requirements.txt* przy użyciu komendy: **pip install -r requirements.txt** oraz pobrać resztę repozytorium.

Testy

W przypadku pretrenowania nadzorowanego proporcje zbiorów danych uczącego, walidującego i testowego wynoszą 8:1:1. Do obliczania wartości **F1** używany jest zbiór testowy, który nie jest podawany do modelu przed końcem uczenia.

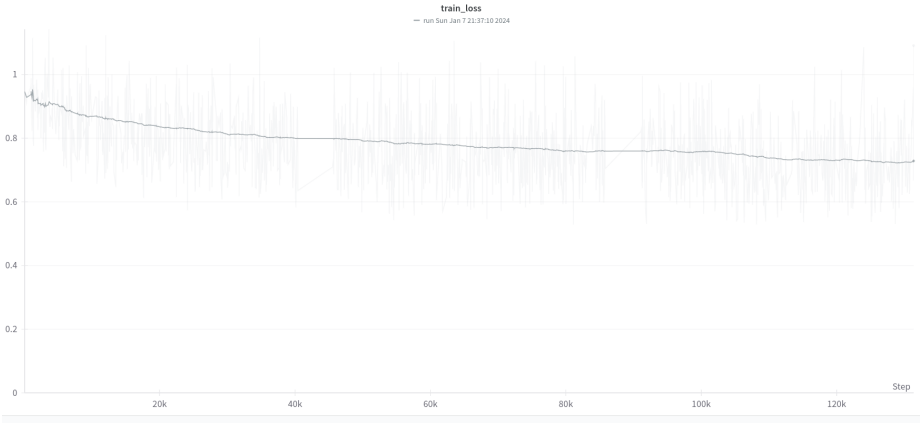
Początkowe testy przeprowadziliśmy dla domyślnego współczynnika uczenia $lr = 0.001$, co okazało się być za dużą wartością i sprawiało, że model niczego się nie uczył.

Rysunek 1: Przebieg pięciu epok uczących dla $lr = 0.001$

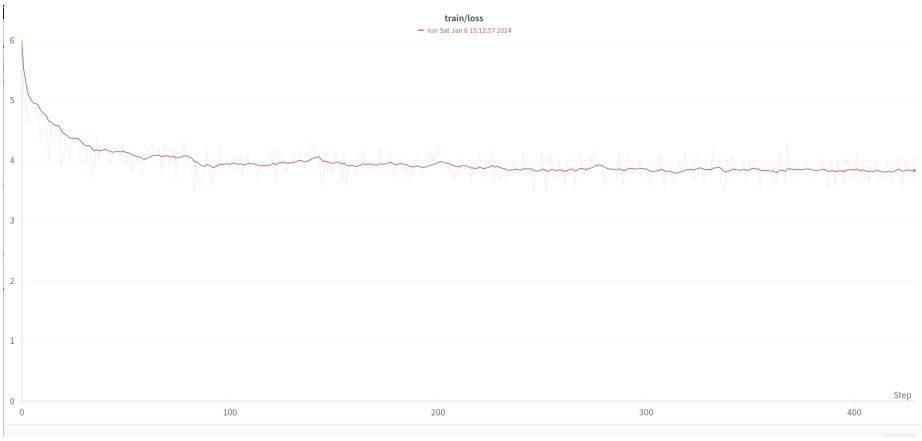
Ostateczne eksperymenty wykonaliśmy dla następujących hiperparametrów:

- Ilość epok uczących na zbiorze QQP: 3
- Ilość epok uczących na podzbiorach iSTS: 20
- Współczynnik uczenia: 10^{-5}
- Algorytm optymalizacji: AdamW

Uzyskując następujące wykresy (ciemniejsza funkcja jest wygładzeniem prawdziwego, przezroczystego przebiegu):

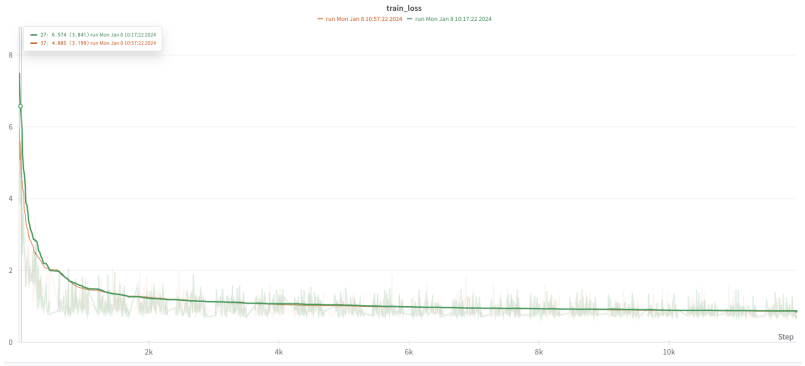


Rysunek 2: Błąd na zbiorze uczącym podczas pretrenowania nadzorowanego.

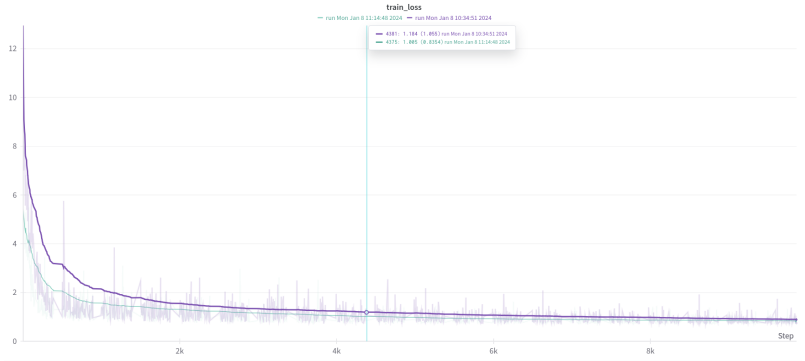


Rysunek 3: Błąd na zbiorze uczącym podczas pretrenowania nienadzorowanego.

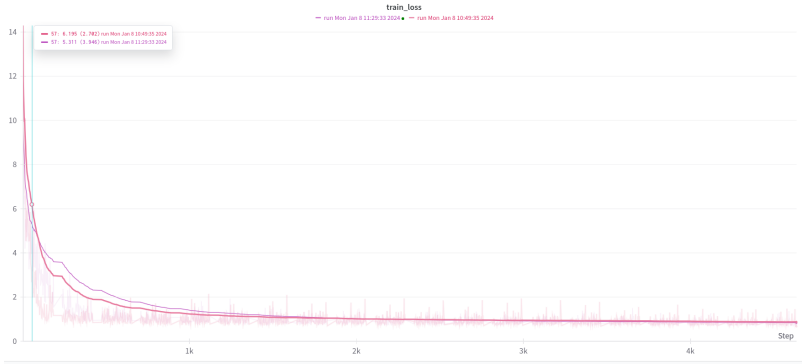
Na każdym z trzech poniższych wykresów możemy zaobserwować 2 niezależne przebiegi. Są to wartości funkcji straty podczas fine-tuningu po uczeniu nadzorowanym i nienadzorowanym, na kolejnych wykresach są to odpowiednio: rys. 4 zielony i czerwony, rys. 5 fioletowy i turkusowy, rys. 6 różowy i fioletowy. We wszystkich trzech przypadkach pretrenowanie nienadzorowane gwarantowało mniejszy błąd, z którego zaczynał się fine-tuning. Po ok. 7 epokach wartości obu błędów wyrównywały się. Podobnie jak w projekcie [3] funkcja straty po ustabilizowaniu się dla każdego podzbioru oscylowała wokół wartości 1.



Rysunek 4: Przebieg błędu podczas fine-tuningu na podzbiorze *images*.



Rysunek 5: Przebieg błędu podczas fine-tuningu na podzbiorze *headlines*.



Rysunek 6: Przebieg błędu podczas fine-tuningu na podzbiorze *answers-students*.

Wyniki

Poniższe tabele ukazują końcowe wyniki F1 otrzymane po obu rodzajach pretrenowania i fine-tuningu (wartości zwrócone przez skrypty *evalF1_penalty.pl* i *evalF1_no_penalty.pl* są takie same z wyjątkiem miary *F1 Typ+Sco*).

Zbiór	**images**	**headlines**	**answers-students**
F1 Ali	1.0000	1.0000	1.0000
F1 Type	0.5116	0.5490	0.6157
F1 Score	0.8469	0.8698	0.9026
F1 Typ+Sco (penalty)	0.7184	0.7048	0.7129
F1 Typ+Sco (no penalty)	0.4896	0.5341	0.6093

Tabela 1: Wyniki dla pretrenowania nadzorowanego

Zbiór	**images**	**headlines**	**answers-students**
F1 Ali	1.0000	1.0000	1.0000
F1 Type	0.5016	0.5817	0.6157
F1 Score	0.8457	0.8717	0.9175
F1 Typ+Sco (penalty)	0.6990	0.7001	0.7055
F1 Typ+Sco (no penalty)	0.4671	0.5504	0.5909

Tabela 2: Wyniki dla pretrenowania nienadzorowanego

Porównanie z rozwiązaniami dotychczasowymi

Koledzy Szaknis, Kulus i Strykowski [3] wspominają, że nie wiedzieli, czy miara *F1 Typ+Sco* obliczona w poprzednich iteracjach projektu dotyczy wyników *penalty* czy *no penalty*, dlatego wartość ta została skopiowana do obu tabel dla projektów, w których jej pochodzenie nie było jasne.

Dataset	F1 score								Nasz	
Solution	Net Mazese_Grud	t Grudkc	a Konop	RTa Kacz	T5	RTa Budz	LNet Kulu		Nadzorowany	Nienadzorowany
Images	0,910	0,880	0,936	0,950	0,659	0,828	0,952	0,866	0,847	0,846
Headlines	0,912	0,889	0,936	0,953	0,724	0,867	0,955	0,878	0,870	0,872
Answers	0,919	0,924	0,949	0,956	0,705	0,953	0,958	0,896	0,903	0,918

Dataset	F1 type								Nasz	
Solution	Net Mazese_Grud	t Grudkc	a Konop	RTa Kacz	T5	RTa Budz	LNet Kulu		Nadzorowany	Nienadzorowany
Images	0,766	0,626	0,754	0,649	0,753	0,405	0,662	0,511	0,512	0,502
Headlines	0,744	0,684	0,743	0,634	0,720	0,511	0,616	0,562	0,549	0,582
Answers	0,801	0,771	0,835	0,764	0,761	0,853	0,714	0,616	0,616	0,616

Dataset	F1 score+type (penalty)								Nasz	
Solution	Net Mazese_Grud	t Grudkc	a Konop	RTa Kacz	T5	RTa Budz	LNet Kulu		Nadzorowany	Nienadzorowany
Images	0,729	0,743	0,812	0,732	0,611	0,635	0,605	0,747	0,718	0,699
Headlines	0,713	0,743	0,812	0,767	0,646	0,638	0,602	0,712	0,705	0,700
Answers	0,755	0,782	0,847	0,801	0,674	0,861	0,699	0,714	0,713	0,706

Dataset	F1 score+type (no penalty)								Nasz	
Solution	Net Mazese_Grud	t Grudkc	a Konop	RTa Kacz	T5	RTa Budz	LNet Kulu		Nadzorowany	Nienadzorowany
Images	0,729	0,743	0,812	0,732	0,611	0,635	0,605	0,495	0,490	0,467
Headlines	0,713	0,743	0,812	0,767	0,646	0,638	0,602	0,523	0,534	0,550
Answers	0,755	0,782	0,847	0,801	0,674	0,861	0,699	0,611	0,609	0,591

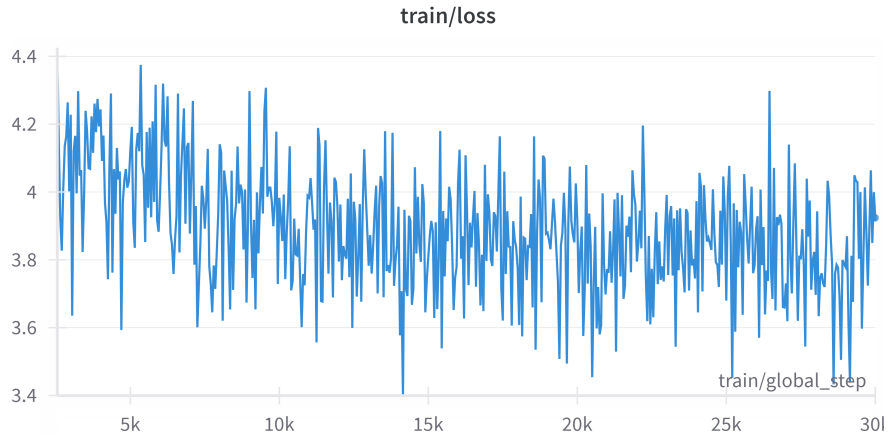
Rysunek 7: Porównanie wyników F1 dla pretrenowania nadzorowanego i nienadzorowanego z poprzednimi iteracjami projektu.

Niestety osiągnięte przez nas wyniki należą do jednych z gorszych prawie w każdej kategorii. Nie da się również jednoznacznie stwierdzić, które podejście (pretrening nadzorowany czy nienadzorowany) daje lepsze rezultaty, co jest również uzewnętrznione na rys. 4, 5 i 6.

Wnioski

Główną trudność w obszerniejszym przebadaniu wpływu hiperparametrów na końcowe wyniki modelu sprawiła konieczność posiadania dobrego sprzętu (tj. karty graficznej). Po wykorzystaniu wersji próbnej Google Colab 1 epoka przy uczeniu nadzorowanym estymowana była na 5 godzin, a epoka uczenia nienadzorowanego na 20 godzin, dodatkowo Google nie daje gwarancji, że jakiekolwiek darmowe jednostki obliczeniowe będą zawsze dostępne. Nie dawało nam to pola do przeprowadzenia jakichkolwiek eksperymentów, dlatego byliśmy zmuszeni do wykorzystania prywatnej mocy obliczeniowej, która na szczęście okazała się wystarczająca.

Krytycznym hiperparametrem okazał się współczynnik uczenia. Przy wcześniej wspomnianych testach z $lr = 0.001$ okazało się, że taka wartość współczynnika jest za duża. Przetestowaliśmy również współczynnik $lr = 10^{-6}$, który z kolei okazał się zbyt mały. Rys. 8 przedstawia przebieg części epoki uczenia nienadzorowanego. Po obserwowaniu jak błąd na zbiorze uczącym oscyluje wokół 4 przez 30 tys. kroków postanowiliśmy przerwać uczenie i ustawić lr na 10^{-5}.



Rysunek 8: Część przebiegu epoki uczenia nienadzorowanego dla $lr = 10^{-6}$

Porównując metody obie pretrenowania łatwo dojść do wniosku, iż żadna metoda nie jest wprost lepsza od drugiej. Dla każdego ze zbiorów *images, headlines, answers-students* fine-tuning po uczeniu nienadzorowanym startował z mniejszej wartości błędu, natomiast nie gwarantowało to szybszej zbieżności do minimalnej wartości (rys. 6). Niestety żadna z tych metod nie generuje zadowalających wyników.

Literatura

[1] Agirre et al. *SemEval-2016 Task 2: Interpretable Semantic Textual Similarity*. June 2016. URL: https://aclanthology.org/S16-1082/.

[2] X. Liang. *What is XLNet and why it outperforms BERT*. Jan. 2019. URL: https://towardsdatascience.com/what-is-xlnet-and-why-it-outperforms-bert-8d8fce710335.

[3] Michał Szaknis, Rafał Kulus, and Jakub Strykowski. *Projekt NLP XLNet_M2*. May 2022. URL: https://github.com/L0czek/Mazury/tree/master.

[4] Y. Yang et al. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. Jan. 2020. URL: https://arxiv.org/abs/1906.08237.