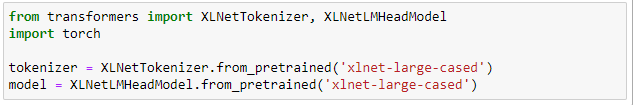
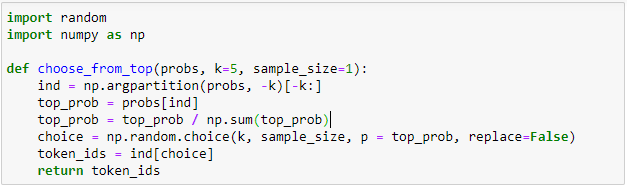
NLP BEADANDÓ

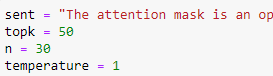
Benyovszki Balázs Zoltán, Rácz András István



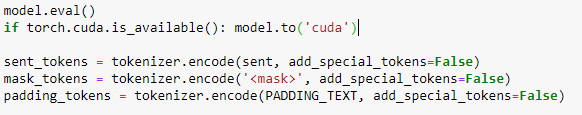
XLNetet használtunk a beadandó project elészítéséhez. Ez arra jó, hogy egy előre betanított modellt tölt le a számítógépünkre ezért nem kell arra erőforrást fordítanunk, hogy betanítsuk a modellt. Mi az [xlnet-large-cased at main (huggingface.co)](https://huggingface.co/xlnet-large-cased/tree/main) tanuló adathalmazt használtuk. Az XLNet Large English model 24-layer-ből, 1024-hidden rétegből, 16-heads-ből, 340M paraméterből áll. Ugyan ilyen módszerrel töltjük le a xlnet tokenizálóját is. A projeknek szüksége van egy PADDING\_TEXT és a program amikor generálja a szöveget akkor megpróbál minnél jobban hasonlítani erre mondatszerkezet és alak téren.



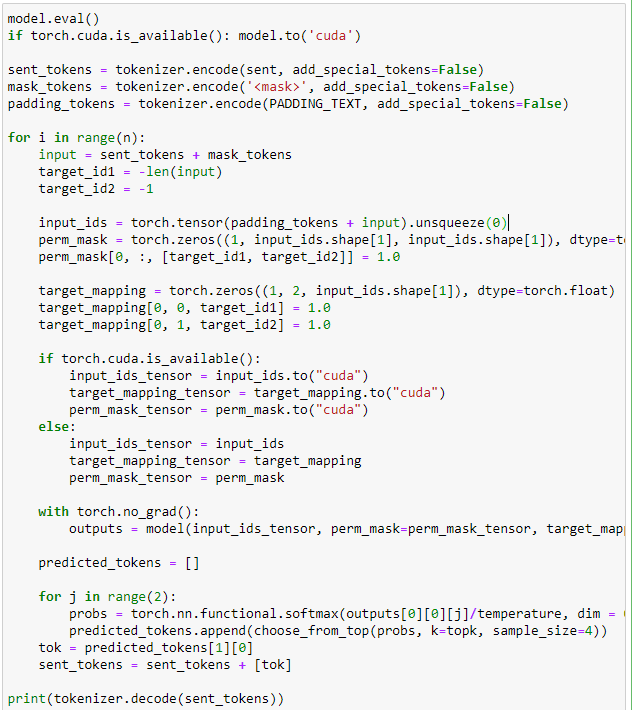
Itt látható a topk módszer implementációja ezzel azt érjük el, hogy azok a szavak amik nagy valószínüséggel nem kerülnének bele a mondatba azokat kiszűrje. Indirect partícionálást végzünk a bemeneti valószínüségek tömbjén ezt a k szerint fogja megtenni. Utána ennek a listának vesszük az utolsó k elemét. Ezek leszenk a toppos szavaink ezeknek tehát nagy valószínüséggel jól beleillenek a mondatba. Utána ezt a listát normalizáljuk tehát elosztjuk a lista összegével. Utána random válsztunk 1 et a numpy segítségével. A random választ egy számot a k ból. Ezt a választást befolyásolja a p opcionális parameter ami azt tartalmazza, hogy melyik számot milyen valószínüséggel kéne választani. Végül választunk egyet majd visszatérünk vele.



Itt az alapot láthatjuk. Ebből a szövegből fog kiindulni a generálás. Utána a topk számát láthatjuk. Ezt fogja használni a felső függvény ezt adjuk át neki. Az n azt jelent, hogy hányszor fog lefutni a ciklus. És a temperature azt adja meg, hogy mennyire legyen biztos a model a kiválasztott szóban.



Kiértékelő módba helyezzük a modellt. Ez azt jelenti, hogy minden rétegünket átrakjuk tanulómodból kiértékelőbe. Mivel az egyik csapattárs rendelkezett erős videókártyával ezért if el eldöntöttük, hogy épp melyikünk futtatja a programot. Utána az xlnet tokenizátorával tokenizáljuk a kezdőmondatot a mascot és a padding textünket ezek jó kis tömbök lesznek.



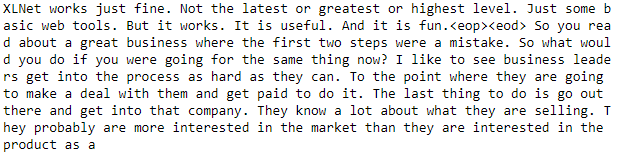
A szöveg generálást egy for ciklusban végezzük. Itt használjuk fel az előbb beállított n változót. A bemenetünk az az alapszóból és a maszkból fog állni. Az 1. target id-nk az e tömb hosszának a -1 szerese lesz. A padding szövegünk tokenjeiből és az inputból tensort készítünk majd ezt n-nek az elejére rakunk egy 0-át. Ezzel fogjuk prediktálni a maskolt tokeneket. Utána készítünk egy 0 lebegőpontos számokból álló mátrixot az input\_ids mintájára. Utána biztosra mmegyünk és beállítjuk, hogy az előző token-nek ne lássák az utolsó tokent. Targetre is készítünk egy 2 dimenziós tömböt. Majd megadjuk az 1. predikcióinkat. Itt szintén ketté válik a kód egy cudásra és egy nem cudásra.

A no\_grad() hasonlóan működik mint az eval() csak gyorsabb.

Ezután a modellünknek megadjuk a megfelelő paramétereket.

Majd bejárjuk a 2 dimenziós tömbünket egy for ciklussal.

A valószínűségeket egy a softmax függvény fogja kiszámolni és egy numpy tömbbe tölti. A prediktált tokenekhez hozzáfűzzük a topk módszer által választott tokent. A kimenethez hozzáfűzzük a tokent és a végén decodeoljuk, hogy megkapjuk az üzenetet.



A generált szöveg minősége javítható ha a szavakat nem egyesével generáljuk, hanem mondjuk szekvenciákat amiből a legvalószínűbbet kiválasztjuk.