Проект по “Откриване на знания в текст”

Работа по задача 1 от CLEF2022-CheckThat!

Разпознаване на релевантни твърдения в туийтове.

Иван Арабаджийски

Резюме

Курсовият ми проект по “Откриване на знания в текст” се състои в работа по задача 1-English и 1-Bulgarian от състезанието “CLEF2022-CheckThat!”. Целта на задачата е да определи кои от дадените постове от туитър (туийтове) на тематика COVID-19 изразяват релевантни твърдения. Задачата е разделена на четири подзадачи, които целят да отговорят на въпросите “заслужава ли си фактологията в твърдението да се проверява?”, “съдържа ли фактологично твърдение, което може да се провери?”, “твърдението вредно ли е за обществото?” и “очакваме ли да привлече вниманието на политиците и защо?”. Първите три са двоична класификационна задача, т.е. Очакваме отговор “да” или “не”, а последната е мулти-клас класификационна задача. Подходът, който съм избрал за решаване на задачата, е модел, базиран на транформъри, трениран върху данните.

Въведение

С напредване на времето, темата за пандемията от COVID-19 бива коментирана в социалните мрежи все повече и повече. Естествено, социалните мрежи са свободна платформа, в която всеки човек може да изразява мнението си. Така нямаме подходящ начин да отсеем правдоподобните и релевантни твърдения в изказванията на хората. Това поражда нуждата от състезанието “CLEF2022-CheckThat!”, което дава възможност на учените в областта на обработката на естествен език да споделят различни идеи за решаването на този проблем. В този проект ще преставя своите идеи за решаване на задачата. Тъй като транформърите вече са се превърнали в отвърден метод за решаване на задачи в областта на обработката на естествен език, аз съм решил да ги използвам като основен метод за решаване на задачите. Входните данни, форматът на изходните данни, както и функциите за оценка на резултатите, са предоставени от организаторите на състезанието. Входните данни са разделени на тренировъчен, валидационнен и тестови файл. Тренировъчните данни съдържат темата (винаги COVID-19), идентификационния номер на туийта, неговия линк (url), текста му и класа му. Класовете се определят от числата нула и едно, представящи “да” и “не” в случая на задачи 1A, 1B и 1C и самите класове, описани с думи, в случая на задача 1D. От тях за трениране на модела използваме само текста, защото той съдържа всичката информация. Очакваният изход е за всеки туийт от тестовото множество да предскажем съответния му клас и да го попълним във файл, който после бива оценен от оценяващия скрипт, предоставен от организаторите на състезанието. Той използва различни метрики за оценка на всяка от четирите подзадачи. F1 за положителния клас за 1A и 1C, точност (accuracy) за 1B и претеглено F1 за 1D. Това са и финалните резултати за всяка задача.

Преглед на областта

Предишни опити за решаване на подобни класификационни задачи използват класификатори като Random Forest, SVM, Multinomial Naive Bayes и характеристики, базирани на TF-IDF репрезентации, части на речта, разпознаване на емоции в текста и други. В предишни издания на състезанията могат да се наблюдават подходи, използващи рекурентни дълбоки невронни мрежи за класификатори, в които всеки токен се представя чрез ембединг, чрез части на речта и по други начини. Дори в предишно издание един от най-добрите резултати е постигнат с класификатор, използващ K най-близки съседи и н-грами на входните характеристики. В последните години, отвърден модел за работата с естествен език са трансформърите. Те набират популярност поради иновативния подход към областта и

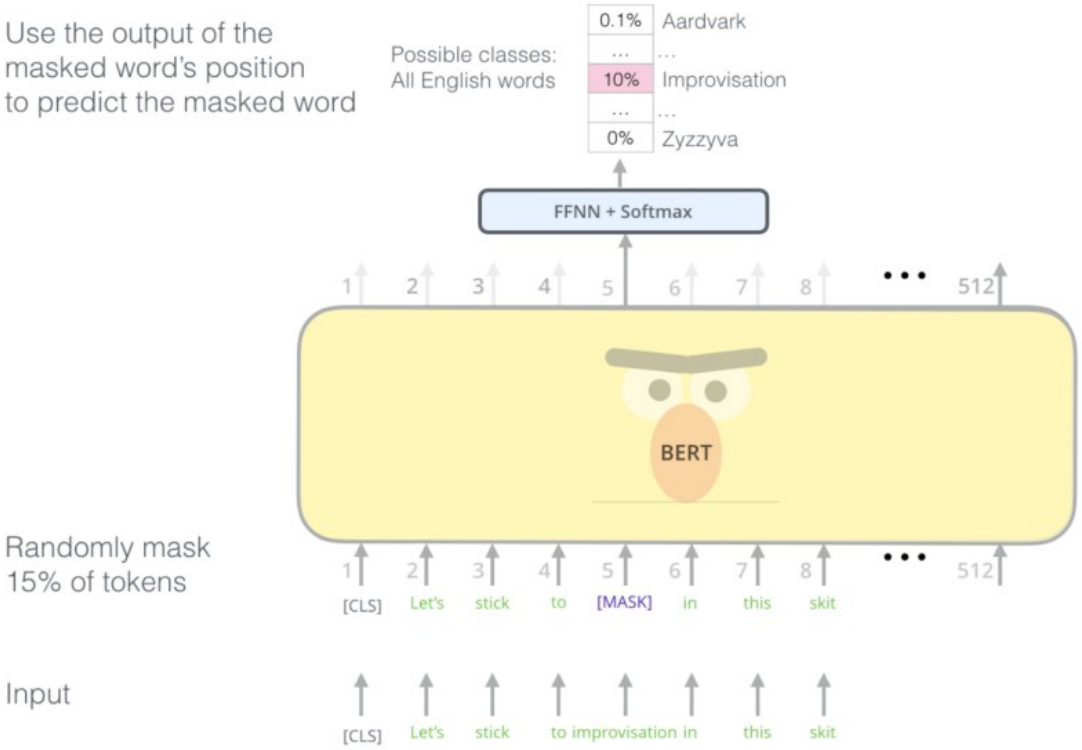
задачата. Най-популярни са BERT, GPT2, GPT3, XLNet, RoBERTa и други. Трансформърите се използват в много задачи, свързани с обработката на естествен език, като машинен превод, класификация на текст, резюмиране на текст, разпознаване на части на речта, предсказване на следващата дума (language modeling), отговаряне на въпроси и т.н. Трансформърът е модел от дълбокото обучение, който приема поредица от символи и ги "превежда" в друга редица, използвайки архитектурата encoder-decoder. Използва механизъм на внимание, за да разпознава най-важните чсти от входните данни. Други модели като рекурентните невронни мрежи използват същия механизъм на внимание, но те са лимитирани от последователното обработване на данните, докато трансформърите, които разчитат само на механизма на внимание, не обработват данните в някакъв конкретен ред. Това им позволява да обработват входните последователности от символи паралелно, което ги прави много по-бързи. На модела предоставяме входните данни под формата на двойки от последователности (sequence pairs) – вход и цел (input and target). В рекурентните невронни мрежи, последователностите се подават токен по токен, като така се предава информацията за позицията на всеки токен. При транформърите, позицията на всеки токен се включва в последователността, която се подава като вход. Енкодерът на модела се състои от стек от n на брой идентични енкодера, всеки от тях има слой на себевнимание и пропагира резултатите напред (feed forward network). Същото се отнася и за декодера, като той има и един слой на внимание енкодер-декодер. Тези слоеве на себевнимание ни позволяват да разпознаем кои токени от входните данни са най-важни в момента на разглеждане на друг токен. Различните слоеве на внимание се състоят от много глави, които се фокусират върху различни части от изречението. Изходите от енкодера се използват за вход на декодера, докато неговят изход минава през един линеен слой. След това, благодарение на софтмакс функция, резултатът е поредица от желания речник. Това е описание на оригиналния труд за транформърите. Естествено, в последствие от него се раждат много различни модели, които правят промени. Например, един от най-успешните BERT (Bidirectional Encoder Representations from Transformers) премахва напълно декодер слоя. Друго важно е, че много от тези модели са предварително тренирани върху голям брой документи. Така те придобиват общо знание за езика. След това прилагаме така наречения метод *transfer learning*.

Данни и характеристики

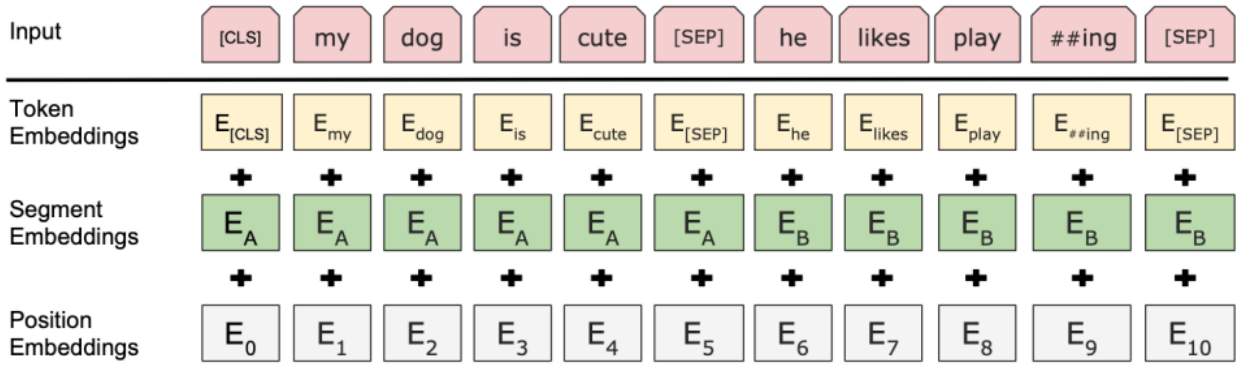
Данните за задачата са предоставени от организаторите на състезанието. Те са колекции от ръчно анотирани туийтове за всяка една от задачите. Предоставени са тренировъчно, валидационно и тестово множество. Тренировъчното множество за задача 1A е с размер от 2123 документа, а тестовото – 149. Размерностите за другите задачи са подобни. Форматът на данните е тема (COVID-19 за всеки), номер на туийта, линк към туийта, неговото съдържание и класът, към който принадлежи. Тестовото множество е предоставено без класа, към който принадлежи, като целта на задачата е *т*ой да се предскаже. Естествено, имаме и тестовото множество още веднъж, този път с правилните класове. Него използваме, за да оценим свършената от модела работа. Като входни данни използваме само текста на туийта - както при обучение и валидация, така и при предсказване. Другата част от характеристиките не допринасят за обучението. За финалните резултати при решаването на тази задача, не правя предварителна обработка на данните. Възможни предварителни обработки, които са опитвани при различни конфигации по време на разработка, са преминаване към малки букви, замяна на всичките линкове със специална дума "url", премахване на тагове и емотикони, използване само на първите 128 символа от всеки туийт. Последното е наложителна обработка, разбира се, ако използваме модели, които могат да работят само с последователности от символи с дължина до 128. Предоставените данни са сравнително чисти и не се е налагала сериозна обработка. Други възможни експерименти са увеличаване на входните данни, като например взимане на 2x положителни и веднъж отрицателни примери или други подобни вариации. Също, при наличието на голямо множество от подходящи данни, свързани с COVID-19, допълнителното претрениране на модела би бил добър експеримент. В противен случай не се знае дали моделът е бил трениран във време, в което вече е имало COVID-19 пандемия, и дали знае изобщо какво е това. В такъв случай, добра идея би било заменянето на изрази и словосъчетания като *ковид 19*, *коронавирус* и други с "ебола". Така поне моделът ще знае, че това е вид болест.

Методи

За реализирането на проекта се използва python и библиотеката pytorch. За трансформър моделите се използва библиотеката transformers. Изработеният от мен модел за решаването на тази задача е сравнително прост: използва трансформър, един дропаут слой за предотвратяване на оувърфитинг, и един линеен слой, който взима изхода на транформъра (тензор с дължина 768) и го смачква до вектор с дължина броя на класовете. Накрая използвам ReLU активация и получавам резултат. Ако вземем argmax от този резултат, ще получим предсказания клас, естествено, под формата на индекс. Затова предварително трябва да направим два речника: клас към индекс и индекс към клас. За щастие, за първите три задачи това не е напълно необходимо, защото класовете, които предсказваме, са анотирани като 0 – не и 1 – да. Това е доста удобно, когато работим със споменатия модел. Основните трансформъри, с които съм правил експерименти, са BERT, BerTweet и RoBERTa (A Robustly Optimized BERT Pretraining Approach). BERT е претрениран без учител - с данни от Wikipedia, като неговата цел е единствено да бъде дотрениран с данните от конкретната задача. По този начин, след базовата тренировка на модела, той има базова идея за езика. Той е претрениран с две конкретни задачи: Masked Language Model и предсказване на следващото изречение. Езиков модел (language model) е всъщност задачата за пресказване на следващата дума по даден контекст. Принципът с маскирането доразвива тази идея. Произволно се маскират (т.е. се заменят със специалния символ “[MASK]”) 15% от думите, след което моделът се опитва да предскаже кои думи са били маскирани.

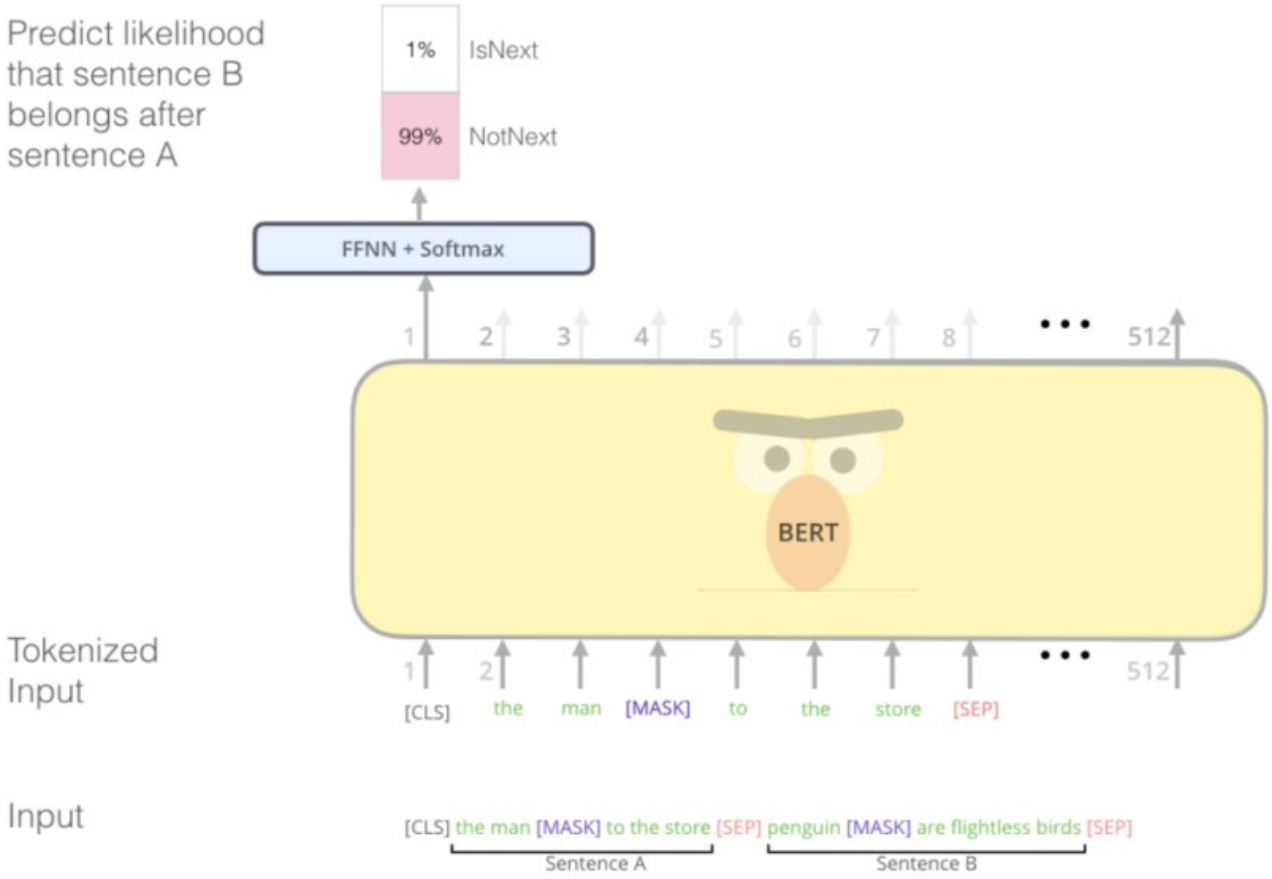


Например, имаме “the man [MASK] to the store” вместо “the man went to the store”. Това позволява на модела да види думите от двете страни на маската, което му позволява да учи контекст в двете посоки, а не само в едната. Това му помага и да разбере дали две изречения са контекстно свързани. Ако изречение A е “[CLS] the man [MASK] to the store”, а изречение B е “penguin [MASK] are flightless birds [SEP]”, BERT ще може да класифицира дали изречение B е продължение на A или не. BerTweet е модел с архитектурата на BERT, който е претрениран с 850 милиона туийта на английски език. При използването на тези модели е важно предварително текстът да се токенизира. BERT работи с три вида ембединги: ембединги на токените, ембединги на сегментите и позиционни ембединги.



Първоначално всички специални токени и думи в речника се преобразуват до индекси. Например, ако изречението е “The cat is walking. The dog is barking”, тогава то трябва да се преобразува по следния начин: “[CLS] the cat is walking [SEP] the dog is barking”. След това, до индекс в нашия речник. Така изречението ще изглежда по следния начин: “[1, 5, 7, 9, 10, 2, 5, 6, 9, 11]”. Нека не забравяме, че 1 и 2 са съответно [CLS] и [SEP]. Сегментен ембединг обикновено разделя две изречения едно от друго. Те обикновено се определят като 0 и 1. Позиционният ембединг запазва информация за наредбата на думите в изречението. За токенизиране на входните данни съм използвал tokenizer от библиотеката transformers. От същата библиотека съм използвал и самите претренирани трансформър модели. Библиотеката pandas съм използвал за представяне на данните (а именно DataFrame).

Реализирал съм собствени помощни класове Dataset и BertClassifier. BertClassifier приема като параметър трансформъра, който да използва. По този начин всички експерименти могат да се оправляват само от няколко параметъра. За лос функция съм използва крос ентропия. H(P, Q) = — sum x in X P(x) * log(Q(x)). За оптимизатор от pytorch – Adam. Това е разширение към стохастичното градиентно спускане.



Резултати

Използваните от мен хиперпараметри са learning rate = 1e-6, размер на бача = 2. Метриките за тестване са различни за всяка една от четирите подзадачи. Те са определени от организаторите на състезанието, както и са дадени скриптове за тяхното изчисляване. Задачи 1A и 1C се оценяват с метриката F1 с респект към положителния клас (т.е. 1). 1B се оценява с точност (accuracy), а 1D се оценява с претеглен F1. Експериментите включват използване на различни трансформър модели – BERT, RoBERTa, BerTweet, тренирани за различен период от време (епохи) и размер на бача.

Таблица 1
Задача 1A Английски

Model	Epoch	Batch Size	Accuracy	Precision	Recall	F1
bertweet-base	5	2	0.624	0.380	0.692	0.490
bert-base-cased	10	2	0.637	0.377	0.589	0.459

Таблица 1
Задача 1B Английски

Model	Epoch	Batch Size	Accuracy	Precision	Recall	F1
bert-base-cased	15	2	0.721	0.719	0.721	0.719

Таблица 1
Задача 1C Английски

Model	Epoch	Batch Size	Accuracy	Precision	Recall	F1
bertweet-base	15	2	0.792	0.38	0.475	0.422

Таблица 1
Задача 1D Английски

Model	Epoch	Batch Size	Accuracy	Precision	Recall	F1
bert-base-cased	15	2	0.784	0.682	0.784	0.727
bertweet-base	10	2	0.665	0.699	0.665	0.679

Литература

[1] Barla, N. (2022, March 21). *How to code bert using pytorch - tutorial with examples*. neptune.ai. Retrieved July 2, 2022, from https://neptune.ai/blog/how-to-code-bert-using-pytorch-tutorial

[2] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, May 24). Bert: Pre-training of deep bidirectional Transformers for language understanding. arXiv.org. Retrieved July 2, 2022, from https://arxiv.org/abs/1810.04805

[3] D. Q. Nguyen, T. Vu, A. T. Nguyen, BERTweet: A pre-trained language model for English Tweets, arXiv preprint arXiv:2005.10200 (2020).

[4] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, RoBERTa: A Robustly Optimized BERT Pretraining Approach, arXiv:1907.11692 [cs] (2019). URL: http://arxiv.org/abs/1907.11692, arXiv: 1907.11692.

[5] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, R. Soricut, ALBERT: A Lite BERT for Self-supervised Learning of Language Representations, arXiv:1909.11942 [cs] (2020). URL: http://arxiv.org/abs/1909.11942, arXiv: 1909.11942.

[6] V. Sanh, L. Debut, J. Chaumond, T. Wolf, DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter, arXiv:1910.01108 [cs] (2020). URL: http://arxiv.org/abs/1910.

[7] E. Williams, P. Rodrigues, V. Novak, Accenture at CheckThat! 2020: If you say so: Post-hoc fact-checking of claims using transformer-based models, arXiv:2009.02431 [cs] (2020). URL: http://arxiv.org/abs/2009.02431, arXiv: 2009.02431.

[8] Hugging face – the AI community building the future. Hugging Face –. (n.d.). Retrieved July 2, 2022, from https://huggingface.co/

[9] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019, July 26). *Roberta: A robustly optimized Bert pretraining approach*. arXiv.org. Retrieved July 2, 2022, from https://arxiv.org/abs/1907.11692

[10] Winastwan, R. (2021, November 10). *Text classification with Bert in Pytorch*. Medium. Retrieved July 2, 2022, from https://towardsdatascience.com/text-classification-with-bert-in-pytorch-887965e5820f