Evaluace logické podobnosti dvojice vět

David Omrai
ČVUT - FIT
omraidav@fit.cvut.cz

30. prosince 2021

1 Úvod

Tato práce se zabývá problémem rozhodnutí vztahu dvojice vět mezi sebou. A to zdali informace obsažené v jedné nejsou ve sporu s těmi obsaženými ve druhé. Jedná se tedy o problém NLP (Natural Language Processing), ve kterém extrahuji informace z textu tak, aby s tím dále mohl počítač pracovat a vyhodnotit vztah mezi nimi.

Pro přiblížení problému této práce zmíním tři možné vztahy dvojice vět, které mohou nastat.

- **Vyplývá** - informace obsažená v první větě souhlasí s informací ve druhé
- **Kontradikce** - informace obsažená v první větě vyvrací informaci ve druhé
- **Beze vztahu** - informace v obou větách spolu nikterak nesouvisí

Věty dále v textu nazývám *premisa* a *hypotéza*. *Premisa* reprezentuje výrok, který je daný. *Hypotéza* je naopak tvrzení, o kterém nevíme zdali vyplývá z *premisy*.

2 Studium problému

Úkolem tohoto úkolu tedy je rozhodnout, jaký je vztah mezi hypotézou a premisou v datech. Jelikož během využívání modelu nelze používat tyto slovní označení, tak se označují trojicí čísel 0, 1 a 2.

V současné době je pro řešení NLI (Natural Language Inferencing) nejčastější využití embeddingů a transformátorů jako je BERT. Jelikož tato práce pro mne slouží k seznámení s touto problematikou, tak jsem nejdříve vytvořil naivní model obsahující veškeré základy s prací nad přirozeném jazyce.

Na začátku práce jsem vycházel z notebooku [5], který tuto problematiku přiblížil pomocí práce s BERTem. Avšak přístup byl spíše směřován na rozvíjení vytvořeného modelu než pochopení, co se v pozadí vlastně děje.

Lepší úvod do základů této problematiky přinesl notebook [3]. V něm se představuje tzv. tokenizace, neboli jak řešit problém převodu vět na formát, kterému rozumí počítač. Jedná se tedy o první část práce na vstupních datech, kde se každá věta převede na pole čísel, tzv. tokenů. Dále se článek zmínil a využil embedding vrstvu, která tento vstupní vektor diskrétních tokenů převádí podobně jako word2vec s tím rozdílem, že se i učí.

Tuto práci lze rozdělit na dvě části, vektorizaci vět a jejich následná klasifikace. V první části dochází k extrakci informace z vět a je tedy třeba velká pozornost, jakým způsobem se toto provede. Nejznámějším transformerem pro tuto část práce je BERT, proti kterému nelze moc soutěžit, jelikož je naučený na enormním množství dat z mnoha jazyků. Jelikož jsem tento projekt vyvíjel na svém laptopu, který nemá velkou výpočetní kapacitu, abych vytvářel a trénoval model konkurující právě tomuto transformeru, tak se nabídla jiná cesta a to vytvoření vlastního tokenizeru, který se bude učit z dat z úlohy.

S tímto mi pomohl článek [1] pojednávající o různých přístupech pro tokenizaci vstupních dat. Mezi klasické se řadí vytvoření slovníku (full forms), kde se z každého slova stane token, nebo do word pieces, kde se jedno slovo může rozpadnout do několika tokenů, pokud v sobě obsahuje už známá slova.

Po vytvoření svého tokenizeru jsem stál před dalším problémem a to jak tento diskrétní vektor dále převést na spojitý, ve kterém by se projevil vztah jednotlivých prvků. S tímto mi pomohl článek [2]. Ten popisuje o co se vlastně jedná, jak je lze využít v neuronových sítích a jak je se učí. Taktéž pokrývá, že v neuronové síti se nachází tři druhy. Ty co hledají nejbližšího souseda v embedding prostoru se často využívají v doporučovacích systémech, které si zakládají na příslušnosti dat do clusteru kategorií. Dále jako vstup pro modely strojového učení u supervizované úkoly. V závěru taktéž pro vizualizaci konceptů vztahů mezi kategoriemi. Součástí článku se vytváří neuronová síť, která zmíněné přístupy aplikuje. Obsahuje dvě vstupní vrstvy, po kterých následuje první skrytá Dot. Ta je využita po Embedding vrstvě, aby opět vytvořila vektor určitých rozměrů. Model v článku nebyl využit pro klasifikaci, ale část z něj mě inspirovala u návrhu svého modelu pro tuto úlohu.

K finálnímu vytvoření představy podoby mého modelu pomohla série videí [4]. Ta opět rozebrala do hloubky, co je tokenizace a embedding, jak je využít a vytvořit. Videa postupně prezentují vytvoření modelu pro rozpoznávání sarkasmu ve větách.

3 Vstupní data

Jelikož se jedná o výzvu z Kaggle, tak data byla již připravena k použití ve formátu csv. V této soutěži se klasifikují páry vět (premisa a hypotéza) do tří kategorií. Těmi jsou *entailment*, *contradiction* a *neutral*.

Tento dataset obsahuje věty v patnácti různých jazycích. Těmi jsou Arabština, Bulharština, Čínština, Němčina, Řečtina, Španělština, Francouzština, Hindi, Ruština, Swahilština, Thajština, Turečtina, Urdu a Vietnamština.

Celkem je poskytnuta trojice souborů.

- **train.csv** - tento dataset obsahuje ID, premise, hypothesis, label a language.
- **test.csv** - obsahuje totéž co train.csv až na label, který se má predikovat
- **sample_submission.csv** - ukázka v jakém formátu se výsledek po vyhodnocení odevzdává do Kaggle výzvy

4 Metody

V rámci této práce jsem navrhl a implementoval dva modely, kde jeden z nich pokrývá základní přístupy k řešení NLI úlohy a druhý využívá již naučený model BERT.

Pro první model jsem si nejdříve připravil veškeré potřebné části, které využívá ke klasifikaci jednotlivých párů vět. Nejprve je třeba pochopení vět, neboť je nemůžeme jen tak vložit do vektoru. Pro modely, které chceme vytvořil je potřeba získat vektor čísel. První myšlenka tedy je, proč nevyužít reprezentaci slov jakožto čísel. Tento proces se nazývá tokenizace a využil jsem pro něj api *Tokenizer*. Pomocí tohoto api jsem vytvořil funkci, která na vstupu vezme množinu vět a maximální počet slov, které ve vytvořeném slovníku nechat. Tokenizer si definovanou množinu slov pamatuje a ke každému z nich přiřadí číselný token. Pro neznámá slova je využit token <OOV>, neboli neznámé slovo.

Jelikož na svém laptopu nemám tak vysoký výpočetní výkon, tak jsem si z datasetu vzal pouze anglická slova a ty rozdělil na trénovací a testovací množinu. Díky tomu jsem mohl výsledný model otestovat na datech, která nikdy neviděl.

U prvního modelu si nejdříve definuji maximální délku vět a z jaké části se budou slova osekávat či přidávat. Následně si každou větu převedu na vektor tokenů o stejné velikosti a jednotlivé dvojice vět spojím. Tím budu mít umožněno tyto věty vložit vstupní vrstvou do neuronové sítě.

Samotný model je tvořen pěticí vrstev. Vstupní je typu Input s 2*max_sentence_len, neboli odpovídá velikosti vstupních dat. Následuje již zmíněná vrstva Embedding, která převádí vektory tokenů na dense vector. Jelikož je výstupem 3D tensor, tak následující skrytá vrstva je Flatten, která jej upravuje pro další skrytou vrstvu Dense s 42 neurony. Výstupní vrstvou je Dense s 3 neurony a aktivační funkcí softmax, jelikož se jedná o klasifikační úlohu.

Druhý model využívá BERT enkodér jakožto skrytou vrstvu. Jelikož se v datasetu nachází více jazyků tak pro tuto úlohu byla využita bert-base-multilingual-cased. V tomto případě využívám BERT tokenizer, který není naučený pouze na datech z datasetu ale na daleko vetší množině. V čem se tokenizace pomocí BERTa liší od té mé je, že se na začátek dává [CLS] token a mezi každé věty [SEP]. Tyto tokeny umožňují modelu správnou práci s daty.

Jelikož BERT využívá tři druhy vstupních dat, tak jsem pro převod vstupních dat na správný formát vytvořil enkodér. Ten si bere hypotézu, premisu a tokenizer. Z obou vět vytvoří vektory tokenů a tyto dvě věty spojí. Společně s tímto vektorem vytvoří další dva. Jeden obsahující jedničky a nuly dle toho kde se nachází padding a přidané tokeny a druhý určující typ tokenů.

Aby bylo možné porovnat oba modely nad stejnou množinou dat, tak jsem tento model taktéž trénoval a testoval pouze nad anglickými větami.

Samotný druhý model je tvořen trojicí vstupních vrstev, kde každá je pro jeden vytvořený vektor. Embedding vrstvu v tomto modelu zastupuje vytvořený BERT enkodér. Výstupní vrstvou je pak opět Dense se třemi neurony

5 Výsledky

Každý model jsem testoval nad stejnými daty, aby bylo možné zhodnotit jejich rozdíly. U prvního lze vidět, že po 3 sekundách dosáhl přesnosti na testovacích datech 36.93 %. Vezmeme-li přitom na vědomí, že se jedná o jednoduchý model využívající základních přístupů pro řešení NLP úlohy je tento výsledek docela hezký.

Oproti tomu druhý model využívající BERTa dosáhl daleko lepších výsledků už v první epoše. Z důvodů výpočetní náročnosti jsem byl nucen provést pouze jednu epochu, kde se přesnost na testovací množině dostala na 54.40 %.

```
Epoch 1/30
313/313 [==============================] - 3s 8ms/step - loss: 1.
0988 - accuracy: 0.3413 - val_loss: 1.0931 - val_accuracy: 0.3693
```

Obrázek 1: Trénování prvního modelu

```
70/70 [==============================] - 7936s 114s/step - loss: 1.
0736 - accuracy: 0.4199 - val_loss: 0.9701 - val_accuracy: 0.5440
```

Obrázek 2: Trénování druhého modelu

6 Závěr

Tuto semestrální práci jsem využil k seznámení se neuronovými sítěmi více do hloubky při řešení NLP úloh. K pochopení této problematiky jsem vytvořil model využívající základy a také pokročilejší model, který má tyto části v sobě již implementované. Tento projekt půjde dále rozšiřovat a využít jej k implementaci dalších modelů, jejich pochopení a zkoumání rozdílů.

Reference

[1] James Briggs. How to build a wordpiece tokenizer for bert. online, 2021. [cit. 2021–30–12] https://towardsdatascience.com/how-to-build-a-wordpiece-tokenizer-for-bert-f505d97dddbb.

[2] Will Koehrsen. Neural network embeddings explained. online, 2018. [cit. 2021–30–12] https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526.

[3] Tanul Krishnansh Singh. Deep learning for nlp zero to transformers bert. online, 2020. [cit. 2021–30–12] https://www.kaggle.com/tanulsingh077/deep-learning-for-nlp-zero-to-transformers-bert.

[4] TensorFlow. Natural language processing (nlp) zero to hero. online, 2020. [cit. 2021–30–12] https://www.youtube.com/playlist?list=PLQY2H8rRoyvzDbLUZkbudP-MFQZwNmU4S.

[5] Ana Sofia Uzsoy. Tutorial notebook with bert. online, 2020. [cit. 2021–30–12] https://www.kaggle.com/anasofiauzsoy/tutorial-notebook.