Софийски университет „Св. Кл. Охридски”

Факултет по математика и информатика

Курсов Проект

на тема: „Разпознаване на туйтове относно бедствия”

Студент: **Георги Асенов Стаменов** Ф.Н. **26126,**

Курс: „ИИОЗ, Втори Курс“, Учебна година: 2020/21

Преподаватели: **д-р Георги Георгиев, д-р Преслав Наков, проф. Иван Койчев**

==========

Декларация за липса плагиатство:

- Плагиатство е да използваш, идеи, мнение или работа на друг, като претендираш, че са твои. Това е форма на преписване.
- Тази курсова работа е моя, като всички изречения, илюстрации и програми от други хора са изрично цитирани.
- Тази курсова работа или нейна версия не са представени в друг университет или друга учебна институция.
- Разбирам, че ако се установи плагиатство в работата ми ще получа оценка “Слаб”.

10.7.21 г. Подпис на студента: Г. Стаменов

Съдържание

1 Увод

Целта на курсовия проект е да се достигне възможно най-висок резултат в състезанието организирано от онлайн платформата – *Kaggle* , относно разпознаване на туийтове, в които има двусмислени думи и изрази относно бедствия. Целта е да се приложат уменията придобити по време на курса и участие в истинско състезание.

2 Преглед на областа (Извличане на информация от текстови съобщение и метаданните им)

Първия етап е разглеждане на данните и тяхното „изчистване„, което включва премахването на емотикони, нормализирането на думите, премахване на линкове и тн.

Вторият етап е работата с данните и използването на класически модели за машинно самообучение, които ще използваме за отправна точка, както и *BERT*, за целта на което, трябва да се токенизират данните, за да може да работи с тяхната енкодната репрезентация[2].

3 Проектиране

Данните се извличат от предоставения датасет от *kaggle*, след което се обработват чрез програмния език *python*,и негови библиотеки. Ще използвам класическите методи за машино самообучение – Наивен Бейсов Класификатор, Логистична регресия, *SVM* и К близки съседа. След това ще използваме претрениранят моделът *BERT*, който приема токенизираните данни и спрямо тях се досамообучава.

По време на изготвянето на процеса намерих статия[1] изследваща дали има смисъл да се лематизират думи на английски език, тъй като е с по-проста структура. На база тази статия ще сравним експериментите с думи в оригинална и трансоформирана форма.

4 Реализация, тестване/експерименти

4.1 Използвани технологии, платформи и библиотеки

Python, Google Colab Notebook, emoji,nltk, pandas, pyspellchecker, transformers, sklearn, torch, ipywidgets, nltk, kaggle

4.2 Реализация/Провеждане на експерименти

С помощта на програмния език *Python* и библиотеките – *re, spellchecker* и *emoji* почистваме данните от ненужни препинателни знаци, емотикони и проверяваме за сгрешени думи и оформяме данните, така че да са само с малки букви.

С вече почистените данни можем да започнем с класическите модели, чиито резултати можем да видим в таблица 1:

Алгоритъм	*Accuracy*	*Precision*	*Recall*
KNN	68%	77%	68%
Логистична регресия	79%	79%	79%
SVM	77%	77%	77%
Наивен Бейсов класификатор	60%	66%	60%

„Таблица 1“

След това подготвяме данните за модела *BERT* използвайки библиотеката *transformers,* която токенизира чрез *AutoТокенизер* данните. Моделът идва претрениран за английски език, като ние трябва да го *fine tune-м* за конкретната област, в която искаме да може да се ориентира.

Първоначалните резултати не са особено задоволителни – около 45% средна точност, за това трябва да подобрим модела.

Използваме *AdamW* оптимайзер[3] и *loss* функция за да подобрим резултатите на невронната мрежата, което вдига точността от около 45% до около 84%, т.е. почти двойно.

```
Epoch: #1
Training results:
Acc: 0.835, f1: 0.830
Validation results:
Acc: 0.804, f1: 0.799
Epoch: #2
Training results:
Acc: 0.854, f1: 0.854
Validation results:
Acc: 0.801, f1: 0.801
Epoch: #3
Training results:
Acc: 0.893, f1: 0.893
Validation results:
Acc: 0.811, f1: 0.812
Epoch: #4
Training results:
Acc: 0.923, f1: 0.922
Validation results:
Acc: 0.837, f1: 0.836
```

```
Epoch: #5
Training results:
Acc: 0.942, f1: 0.942
Validation results:
Acc: 0.829, f1: 0.829
Epoch: #6
Training results:
Acc: 0.927, f1: 0.926
Validation results:
Acc: 0.824, f1: 0.820
Epoch: #7
Training results:
Acc: 0.928, f1: 0.927
Validation results:
Acc: 0.824, f1: 0.818
Epoch: #8
Training results:
Acc: 0.929, f1: 0.928
Validation results:
Acc: 0.827, f1: 0.822
```

„Резултати за следните параметри:

Скорост на обучение – 0.000027, *Batch* size – 32, Скрити слоеве – 16

Dropout rate – 0.8“

5 Заключение

Класическите модели не предоставят задоволителен резултат, за самото състезание, но се представят относително добре.

BERT се представя по-добре от класическите модели, когато е добре настроен, но все още има място за развитие и не е разгърнал пълния си потенциал.

Бъдещи планове: *AlBERT* и *RoBERTa*

6 Използвана литература

[1] https://www.aclweb.org/anthology/W19-6203.pdf

[2] https://www.kaggle.com/angyalfold/hugging-face-bert-with-custom-classifier-pytorch#PyTorch-setup

[3] https://www.fast.ai/2018/07/02/adam-weight-decay/