МАШИНСКО УЧЕЊЕ:

ИСТРАЖУВАЧКА АНАЛИЗА, МОДЕЛИРАЊЕ, ТРЕНИРАЊЕ, УПОТРЕБА

Меѓународен Славјански Универзитет
„Гаврило Романович Державин”
15.09.2022

Магистерски труд

Студент: Кристина Гацова
Ментор: Александар Петровски

Содржина

1. Вовед 4
2. Мотивација и цел 6
3. Анализа 7
   3.1. Запознавање со податочното множество 7
      3.1.1. Димензии и тип на променливи 7
      3.1.2. Централна Тендеција 8
         3.1.2.1. Мода 8
         3.1.2.2. Аритметичка Средина 9
      3.1.3. Визуелизација 10
         3.1.3.1. Столбест дијаграм 10
         3.1.3.2. Хистограм 12
         3.1.3.3. Квартали и рангови 14
      3.1.4. Варијанса 16
      3.1.5. Стандардна девијација 17
      3.1.6. Кендал Тау 20
4. Дефиниција на модел 20
   4.1. Кластерирање 20
      4.1.1. Мерки за кластерирање 21
      4.1.2. K-means кластерирање 24
      4.1.3. Хиерархиско кластерирање 27
      4.1.4. LDA 27
         4.1.4.1. Демо 28
   4.2. Невронски мрежи 32
      4.2.1. Активациски функции 33
         4.2.1.1. Функција на бинарен чекор 33
         4.2.1.2. Линеарна активациска функција 34
         4.2.1.3. Нелинеарни функции за активирање 35
         4.2.1.4. Сигмоидна / логистичка функција за активирање 35
         4.2.1.5. Tanh функција (хиперболичен тангенс) 37
         4.2.1.6. Функција ReLU 39
         4.2.1.7. Функција ReLU што протекува 41
         4.2.1.8. Параметриска ReLU функција 43

4.2.1.9. Функција на експоненцијални линеарни единици (ELUs) 44
4.2.1.10. Функција Softmax 46
4.2.1.11. Свиш (Swish) 48
4.2.1.12. Гаусова линеарна единица за грешка (GELU) 49
4.2.1.13. Скалирана експоненцијална линеарна единица (SELU) 50
4.2.1.14. Предизвици 51
4.2.1.14.1. Градиенти кои исчезнуваат 52
4.2.1.14.2. Експлодирачки градиенти 52
4.3. Типови невронски мрежи 53
4.3.1. Перцептрон 53
4.3.2. Невронска мрежа со предавање напред 53
4.3.3. Длабока невронска мрежа со предавање напред 54
4.3.4. Рекурентни невронски мрежи 54
4.3.5. Невронски мрежи со продолжена кратка меморија 54
4.3.6. Автоенкодери 55
4.4. Композитни невронски мрежи 56
4.4.1. Енкодер/Декодер 57
4.4.2. Трансформери 59
4.4.3. BART 61
4.4.4. GPT 61
4.4.5. BERT 63
4.4.5.1. Архитектурата на BERT 64
4.4.5.1.1. Моделирање со маскиран јазик 67
4.4.5.1.2. Предвидување на следната реченица 67
4.4.6. ALBERT 69
4.4.7. RoBERTa 69
4.4.8. ELECTRA 70
4.4.9. SpanBERT 70
4.4.10. DistilBERT и TinyBERT 71
5. Машинско учење во облаците 72
5.1. MLaaS – Машинско учење како услуга 72
5.1.1. Amazon Machine Learning и SageMaker 73
5.1.1.1. Машинско учење на Амазон 73

5.1.1.2. SageMaker 74
5.1.2. Microsoft Azure AI платформа 75
5.1.2.1. Azure Machine Learning 75
5.1.2.2. Галерија Azure AI 77
5.1.2.3. Azure Percept 77
5.1.3. Google AI платформа (унифицирана) 77
5.1.3.1. Платформа за вештачка интелигенција (класична) 77
5.1.3.2. Google Cloud AutoML 78
5.1.4. TensorFlow 78
5.1.5. IBM Watson Machine Learning Studio 78
5.1.5.1. Watson Studio и AutoAI 78
5.1.5.2. SPSS моделер 79
5.1.5.3. Невронска мрежа и длабоко учење 79
5.1.6. Споредба на API за машинско учење на Amazon, Microsoft, Google и IBM 79
5.1.6.1. API за обработка на говор и текст: Amazon 81
5.1.6.2. API за обработка на говор и текст: Когнитивни услуги на Microsoft Azure 82
5.1.6.3. API за обработка на говор и текст: Услуги на Google Cloud ML/ Cloud AutoML 83
5.1.6.4. API за обработка на говор и текст: IBM Watson 84
5.1.7. API за машинско учење за видео анализа од Amazon, Microsoft, Google и IBM 85
5.1.7.1. API за обработка на слики и видео: препознавање на Amazon 87
5.1.7.2. API за обработка на слики и видео: Когнитивни услуги на Microsoft Azure 88
5.1.7.3. API за обработка на слики и видео: Услуги на облак на Google/ Cloud AutoML 88
5.1.8. API за обработка на слика и (не) видео: Визуелно препознавање на IBM 89
5.1.9. Специфични API и алатки 89
5.1.9.1. Azure Bot Service 89
5.1.9.2. AWS ML Хардвер 89
5.1.9.3. Bing Search од Microsoft 90
5.1.9.4. Knowledge од Microsoft 90
5.1.9.5. Google Cloud Talent 90
5.1.9.6. Вотсон асистент 91
6. Демо 91
7. Заклучок 107
8. Индекс на користени алатки и литература 108

1. Вовед

Вештачката интелигенција (ВИ) е област на компјутерската наука која се занимава со создавање на интелигентни агенти, кои се системи кои можат да расудуваат, учат и да дејствуваат автономно. ВИ има значителни импликации за секојдневниот живот, вклучително и во областите како што се транспортот, здравството, финансиите и производството. Експлозивниот раст на вештачката интелигенција е еден од најзначајните случувања на нашето време. Напредокот во вештачката интелигенција го трансформира начинот на кој работиме и живееме и помага да се решат некои од најголемите предизвици во светот. Вештачката интелигенција, исто така, има потенцијал да ги направи нашите животи попродуктивни и попријатни, како и да ги зајакне нашите социјални врски и да ги унапреди иновациите и креативноста во нашето општество. Вештачката интелигенција е тесно поврзана со машинското учење. Машинското учење користи алгоритми за да ги анализира податоците и да прави предвидувања врз основа на неговите наоди. Во практична смисла, ова значи дека компјутерите се способни да анализираат и да извлечат заклучоци од големи количини на информации. На пример, алгоритмите за машинско учење се способни да идентификуваат шаблони во множество податоци и потоа да прават предикции врз основа на информациите што ги научиле. Компјутерите можат да користат техники за учење за да донесуваат одлуки и да ги завршат задачите без да бидат експлицитно програмирани да го прават тоа.

Првата личност да го употреби терминот „вештачка интелигенција“ е професорот од Дартмут Џон Мекарти во 1956 година. Сепак, идејата за вештачка интелигенција датира многу подалеку од ова, со филозофи како Рене Декарт и Џон Лок кои зборуваат за можноста за создавање машини за размислување во 17ти век. Терминот „Вештачка интелигенција“ беше измислен од компјутерските научници Џон Мекарти и Херберт Сајмон во 1955 година, како дел од истражувачки проект за потенцијалните апликации на компјутерската технологија. Тие го дефинираа како „наука и инженерство за правење интелигентни машини“. Со текот на годините, полето еволуира за да опфати широк спектар на дисциплини, вклучувајќи математика, психологија, биологија, филозофија и инженерство. Компјутерските научници денес користат напредни техники како што се невронски мрежи и генетски алгоритми за да развијат нови модели на вештачка интелигенција кои се попрецизни и поефикасни од постарите модели. Истражувачите, исто така, работат на начини како вештачката интелигенција да биде подостапна за повеќе луѓе преку развивање системи кои вклучуваат емоционално и социјално разбирање.

Во последниве години, вештачката интелигенција станува сè поприсутна во нашиот секојдневен живот. Може да се најде во сè, од самоуправувачки автомобили и роботски правосмукалки до уреди што се активираат со глас и виртуелни асистенти. Бидејќи овие технологии продолжуваат да се подобруваат и стануваат попристапни, веројатно е дека тие ќе станат уште почести во годините што доаѓаат. Од практична гледна точка, влијанието на вештачката интелигенција врз општеството веројатно ќе биде позитивно. Со автоматизирање на повторувачките задачи и правење сложени пресметки далеку поефикасни, има потенцијал да им го олесни животот на луѓето. Во исто време, важно е да се земат предвид потенцијалните ризици од воведувањето на вештачката интелигенција во општеството. Како што овие машини стануваат поинтелигентни и независни, тие би можеле да претставуваат закана за безбедноста и благосостојбата на луѓето. Исто така, постои можност системите со вештачка интелигенција да бидат хакирани или злоупотребени од луѓе кои

сакаат да ги користат за злонамерни цели. Неодамнешно сознание е дека еден од ВИ моделите кои работи во доменот на здравството успеал самиот по себе да научи да ја детектира расата на човекот само преку рентген снимнки. Колку што е ова интересно, може да биде и опасно доколку се злоупотреби за спроведување расна дискриминација во здравствениот систем во смисла на листа за чекање на органи, хостпитализација и други ситуации каде се работи за распоред на ограничени ресурси. И покрај овие ризици, вештачката интелигенција веројатно ќе има значително позитивно влијание врз човештвото во иднина.

Во понатамошниот текст ќе биде даден осврт на процесот на развој и користење на ML модел. Најпрво ќе биде опишан процесот на анализа на податочното множество. Потоа ќе бидат објаснети неколку од најчесто користените типови модели и ќе биде прикажан практичен пример. По дефиницијата на моделот и негово успешно тренирање, моделот треба да биде достапен за косумација од клиентски апликации. Во поглавјето „Машинско учење во облаците“ ќе биде дадена споредба помеѓу различните cloud сервиси за ML кои во моментов се достапни и најпознати. На крај ќе биде дадено демо во кое чекор по чекор се објаснува процесот на качување на ML модел во облаците.

2. Мотивација и цел

Вештачката интелигенција е интересна и моментално актуелна тема во светот. Во последните неколку години сме сведоци на реална практична примена на вештачка интелигенција со мерлива и опиплива ефикасност. Текстовите генерирани од вештачка интелигенција се до тој степен усовршени што разликуваат стилови на пишување, практично и да не може да се одреди текст напишан од човек и од робот. Предизвик е да се одделат и виртуелните аватари од снимки од реални луѓе.

Во понатамошниот текст ќе биде даден осврт на чекори на имплементација на вештачка интелигеницијa, од анализа на податоци до сервис готов за консумација од клиент. Конкретно, ќе биде даден осврт на NLP (natural language processing) модел, односно модел за процесирање на природен јазик.

3. Анализа

Кога се изработуваат модели за комерцијална употреба има две сценарија:

- Постои проблем кој треба да се реши
- Имаме податочно множество и се обидуваме да најдеме шаблони или поврзаности

Доколку имаме податочно множество за обработка, првиот чекор е запознавања со истото[34].

За демонстрација на процесите ќе користиме податочно множество преземено од Kaggle[9] кое содржи годишна статистика на светско ниво за број на самоубиства, убиства и бруто домашен производ по држава. Множеството на кратко ќе биде адресирано како SHG (suicide, homicide, gdp). За извршување на оваа анализа се користи Python[6] во најголема мера со Pandas[1084]. Pandas е библиотека напишана во Python за анализа и манипулација на податоци.

3.1. Запознавање со податочното множество

3.1.1. Димензии и тип на променливи

Иницијално најдобар начин за овој прв поглед е претставување на податоците во табела. На слика 1 е дадена таква претстава на SHG податочното множество.

	country	iso3c	iso2c	year	Intentional homicides (per 100,000 people)	Suicide mortality rate (per 100,000 population)	GDP (current US$)	GDP per capita, PPP (current international $)	adminregion	incomeLevel
0	Afghanistan	AFG	AF	1970	NaN	NaN	1.748887e+09	NaN	South Asia	Low income
1	Afghanistan	AFG	AF	1971	NaN	NaN	1.831109e+09	NaN	South Asia	Low income
2	Afghanistan	AFG	AF	1972	NaN	NaN	1.595555e+09	NaN	South Asia	Low income
3	Afghanistan	AFG	AF	1973	NaN	NaN	1.733333e+09	NaN	South Asia	Low income
4	Afghanistan	AFG	AF	1974	NaN	NaN	2.155555e+09	NaN	South Asia	Low income

Слика: 1

Обично податочните множества се доста обемни, па добро е да се знае точно колку обемно е актуелното множество. На слика 2 е дадена димензијата на податочното множество.

```
df_shg.shape

(13566, 10)
```

Слика: 2

13 566 редици и 10 колони.

Следно е идентификација на типот на варијаблите (колоните). Спрема вредностите, варијаблите може да се континуирани и дискретни. Во SHG множеството, колоните за идентификација на земјата, административен регион и ниво на примања се дискретни варијабли бидејќи има

конечен број различни вредности кои може да се појават. Останатите се континуирани бидејќи постојат бесконечен број вредности кои може да се појават.

Варијаблите може да бидат:

- номинални, доколку содржат конечен број вредности кои не може да се наредат, пример колоната за административен регион: без разлика дали и како се сортира, редоследот не продуцира никакво дополнително значење.

- ординалните вредности на варијабла се такви да може да се споредат меѓусебе и да се сортираат или рангираат. Пример вредностите за висина на примање кои може да се ниски, средни, високи.

- интервални варијабли се такви каде што меѓу вредностите може да се направи споредба со значење но отсуствува реална нула. Пример, помеѓу 5 и 10; и 10 и 15 степени целзиусови има интервал со значење, но 10 степени не е двапати потопло од 5. Ваков тип на варијабла немаме во актуелниот дата сет.

- варијабли со сооднос се такви што помеѓу вредностите може да се одредат интервали со значење и имаат природна нула. Во SHG множеството GDP е таква варијабла.

3.1.2. Централна Тендеција

Еден од почесто користените концепти во статистиката е централната тенденција. Таа претставува дескриптивна сумаризација на податочното множество преку единствена вредност која го рефлектира центарот на податочната распределба. Заедно со вариабилноста, централната тенденција е гранка од дескриптивната статистика. Централната тенденција може да се опише со медијана, мода или средина.

3.1.2.1. Мода

Модата на една варијабла е вредноста која се појавува најчесто. Таа е мерка за централна тенденција која може да се примени на секаков вид варијабли, но е единствена мерка за номинални варијабли.

Практична примена: каква корисна информација може да се добие од оваа мерка?

Во случајот со SHG множеството, ме интересира за која земја има најмногу статистики. Тоа може да го направиме ако извадиме мода од кодовите на земја. Излезот од командата е даден на слика 3.

```
df["cntrcode"].mode()
```

```
0       AFE
1       AFG
2       AFW
3       AGO
4       ALB
       ...
224     WSM
225     YEM
226     ZAF
227     ZMB
228     ZWE
Length: 229, dtype: object
```

Слика: 3

229 вредности се со најмногу појавувања.

```
df.groupby("cntrcode").size()
```

```
cntrcode
AFE    20
AFG    20
AFW    20
AGO    20
ALB    20
       ..
WSM    20
YEM    20
ZAF    20
ZMB    20
ZWE    20
Length: 229, dtype: int64
```

Слика: 4

Според излезот од слика 4, тоа е случајот.

3.1.2.2. Аритметичка Средина

Најчесто користена мерка за централна тенденција кај варијабли со соодност и интервални варијабли е аритметичка средина. Вредноста се пресметува со равенката:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Практична примена: Во последните 20 години, во Македонија се самоубиваат просечно 9,42 лица на 100 000 жители годишно. Излезот од командата е даден на слика 5.

```
smr_MKD["smr"].mean()
```

```
9.420000000000002
```

Слика: 5

3.1.3. Визуелизација

За поефикасно снаоѓање во податочното множество, корисна е визуелизација на статистики, мерки и вредности. Постојат голем број начини за приказ на корисните информации. Често изборот на график е индивидуална преференца, но постојат неколку најчесто користени т.е. најадекватни прикази во склоп со типот на променливата. Идејата позади визуелизацијата е будење на интуицијата кај корисникот и насочување на истражувањето. За генерирање на дијаграмите ќе ја користиме библиотеката Seaborn[11].

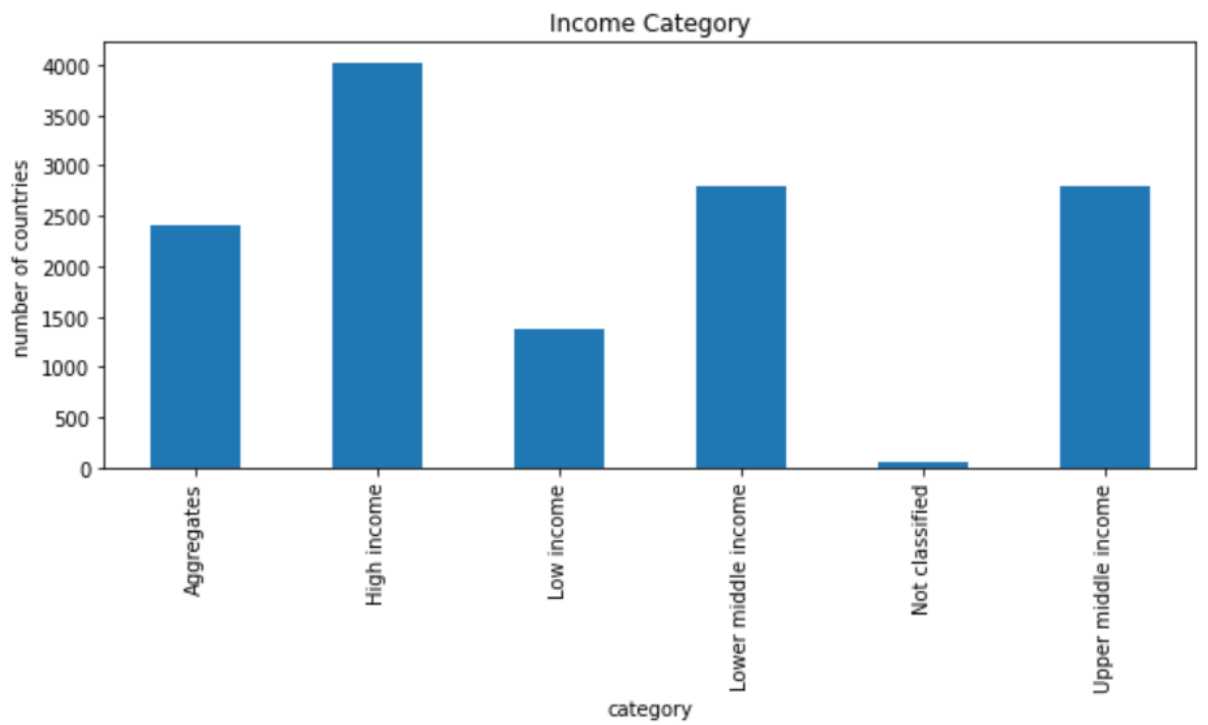
3.1.3.1. Столбест дијаграм

За номинални варијабли, столбестиот дијаграм може да се искористи за приказ на фрекфенцијата на секоја од вредностите. Обично на x оската се поставуваат можните вредности, а на y се фрекфенциите. На слика 6 е прикажан столбест дијаграм на приходот наспроти бројот на земји.

```
dfg = df_shg.groupby(['incomeLevel'])['country'].count()

dfg.plot(kind='bar', title='Income Category', ylabel='number of countries',
         xlabel='category', figsize=(10, 4))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f97c0d80310>
```



Слика: 6

Интересно е што покрај 4те очигледни категории (низок,средно низок, средно висок, висок) и неспецифицираната, има уште една категорија: агрегати. Ова е добар момент да се вратиме назад на податочното множество за да видиме што е тоа агрегат.

```
print((df1.loc[df_shg["incomeLevel"] == "Aggregates"])["country"].drop_duplicates())
```

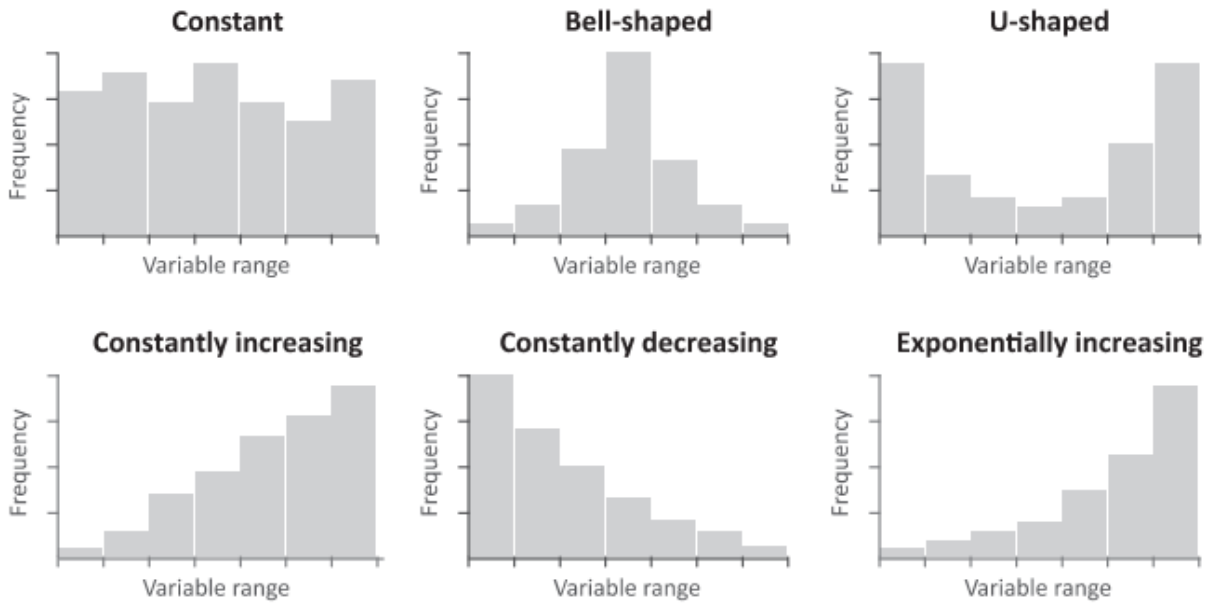
```
81                            Africa Eastern and Southern
132                           Africa Western and Central
489                                            Arab World
2019                               Caribbean small states
2172                         Central Europe and the Baltics
3192                             Early-demographic dividend
3243                                      East Asia & Pacific
3294           East Asia & Pacific (IDA & IBRD countries)
3345          East Asia & Pacific (excluding high income)
3804                                           Euro area
3855                                    Europe & Central Asia
3906          Europe & Central Asia (IDA & IBRD countries)
3957         Europe & Central Asia (excluding high income)
4008                                         European Union
4212              Fragile and conflict affected situations
5130                Heavily indebted poor countries (HIPC)
5181                                          High income
5385                                           IBRD only
5436                                       IDA & IBRD total
5487                                            IDA blend
5538                                             IDA only
5589                                            IDA total
6711                                 Late-demographic dividend
6813     Latin America & Caribbean (excluding high income)
6864     Latin America & the Caribbean (IDA & IBRD coun...
6966          Least developed countries: UN classification
7323                                      Low & middle income
7374                                           Low income
7425                                      Lower middle income
8139                                Middle East & North Africa
8190     Middle East & North Africa (IDA & IBRD countries)
8241     Middle East & North Africa (excluding high inc...
8292                                         Middle income
9159                                         North America
9414                                          OECD members
9516                                   Other small states
9567                             Pacific island small states
10077                              Post-demographic dividend
10128                               Pre-demographic dividend
11046                                        Small states
11250                                          South Asia
11301                                South Asia (IDA & IBRD)
11760           Sub-Saharan Africa (IDA & IBRD countries)
11811          Sub-Saharan Africa (excluding high income)
12984                                      Upper middle income
13392                                               World
Name: country, dtype: object
```

Слика: 7

Според слика 7 очигледно дека податочното множество освен поединечни земји, држи и редици со податоци за региони. Ова е важно да се има во предвид при генерирањето визуелизации и моделирање.

3.1.3.2. Хистограм

За варијабли кои може да се подредат по некво значење и кои имаат голем број на различни вредности, хистограмот е доста адекватна претстава на податоците. Хистограмот се гради на тој начин што вредностите се делат во одреден број кофи. Хистограмот е столбест дијаграм во кој висината на секој столб е бројот на појавувања во една кофа. Со оваа претстава може лесно да се воочи формата на распределбата на одредена варијабла. Постојат неколку вообичаени дистрибуции кои се прикажани на слика 8.

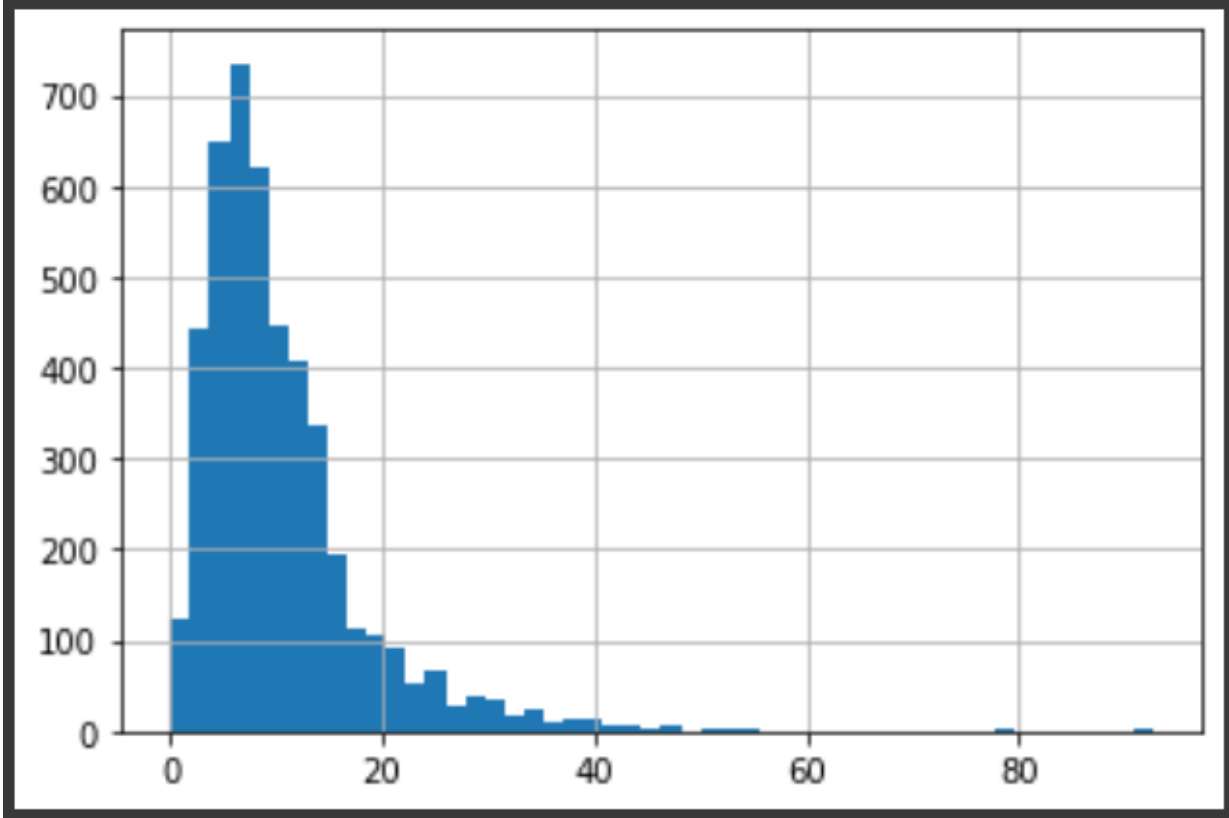


Слика: 8

Вториот график на слика 9 е распределба која најчесто се среќава и уште се нарекува нормална дистрибуција. Идентификацијата на најчесто присутните распределби може да ни помогне во идентификација на аномалии во податоците.

Практичен пример: хистограм на варијаблата suicide mortality rate per 100 000 од SGH множеството.

```
smr_df = df1["smr"]
hist = smr_df.hist(bins=50)
```



Слика: 9

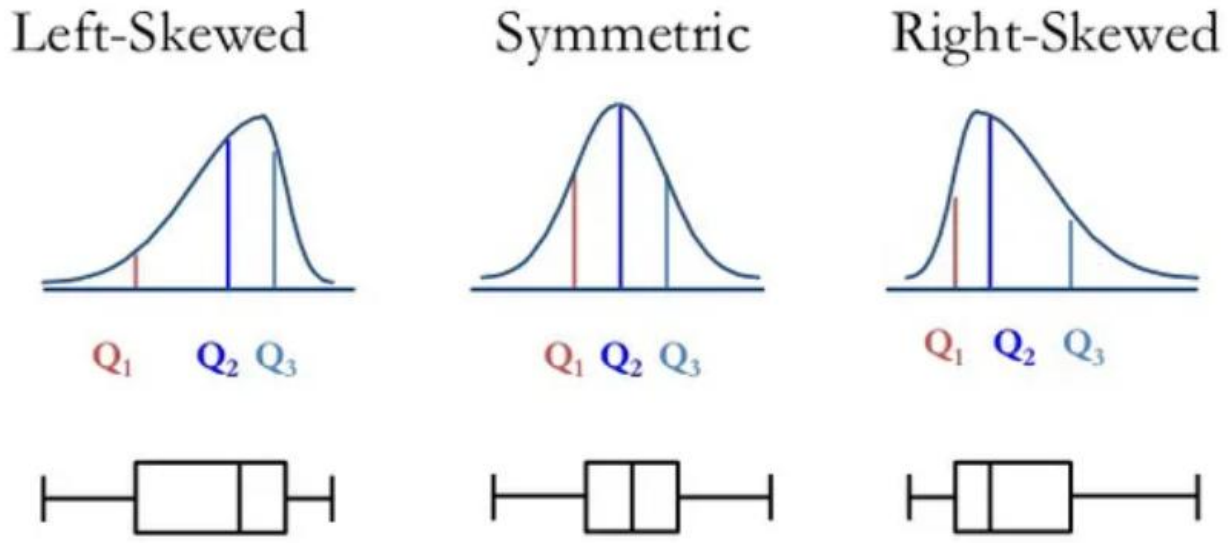
Од дијаграмот јасно се гледа дека варијаблата има опаѓачка распределба. Ова е една од вообичаените распределби, што значи дека податоците најверојатно не содржат значителен број аномалии. Доколку на пример распределбата имаше два врвови, тоа може да биде индикатор за нечисти податоци т.е. дека можеби вредности од две варијабли со слична распределба се ставени во една колона, односно се третираат како единствена варијабла.

Сепак, надвор од главната форма има мал број на вредности кои се невообичаено далеку. Постои шанса ова да се аномалии и треба дополнително да се проверат. Во конкретниов случај, не се работи за аномалии, тоа се статистиките од Лесото која заедно со Гренланд имаат невообичаено висок број на самоубиства. Множеството не содржи статистики за Гренланд.

3.1.3.3. Квартали и рангови

Често користени техники за преглед на податоци од ординална природа се рангот на вредностите, односно најниската и највискоата вредност; и квартали. Кварталите се пресметуваат откако податоците се сортираат со тоа што се наоѓа медијаната и на новодобиените два сегмента се

наоѓаат нивните медијани. Визуелната репрезентација е со дијаграм со квадрати или бокс дијаграм. На слика 10 се прикажани дијаграмите за неколку чести распределби.



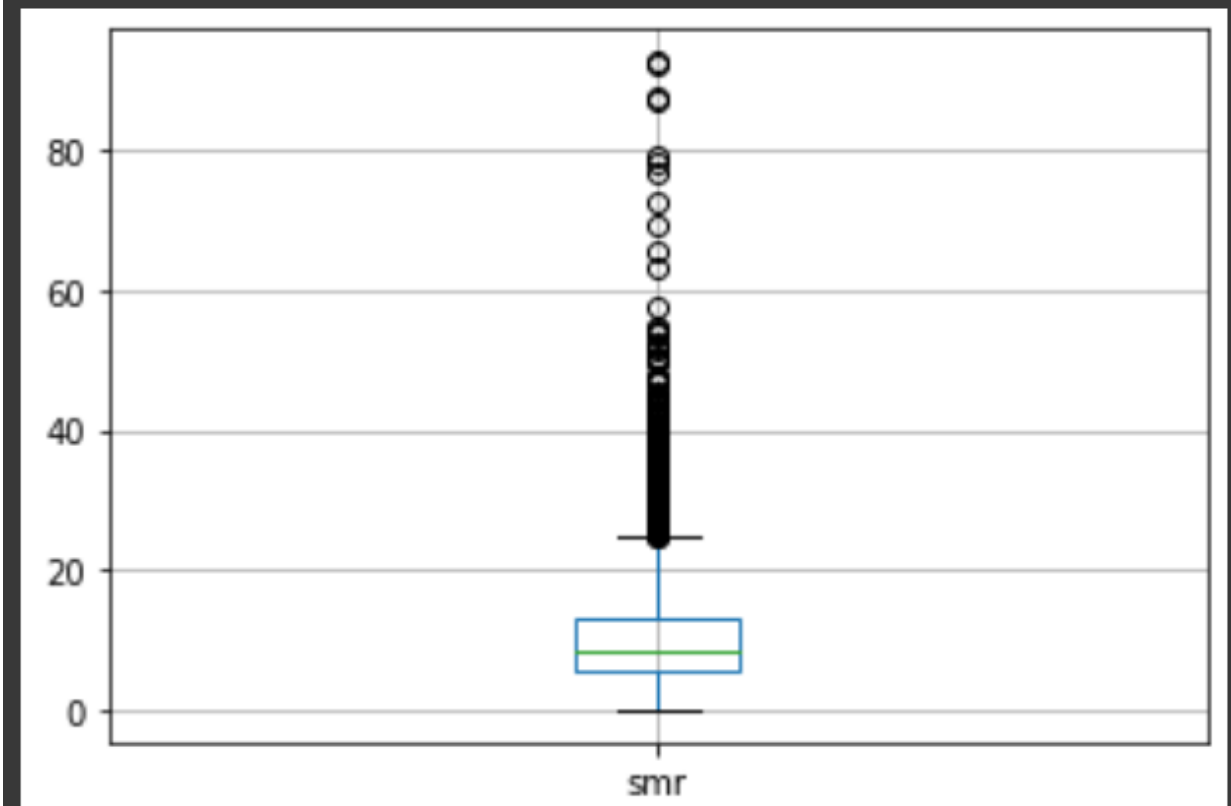
Слика: 10

Крајните вертикални линии се почетокот и крајот на првиот и четвртиот квартал соодветно. Квадратот ги има вредностите од вториот и третиот квартал. Вертикалната линија во квадратот ја одредува медијаната. Во некои случаи може да биде маркирана и аритметичката средина со точка. Почетокот и крајот на првиот и четвртиот квартал се ограничени со максимална должина од 1,5 од интер кварталниот распон, односно интервалот од почетокот на вториот до крајот на третиот квартал. Сите вредности кои се надвор од минимумот и максимумот маркираат со дополнителни точки или кружни маркери.

Предноста на бокс дијаграмот е компактноста на репрезентацијата. Многу прегледно може да се претстават распределбите на повеќе варијабли една до друга и да се воочат разлики или сличности.

Практичен пример: бокс дијаграми на самоубиства од SHG податочното множество. Слика 11.

```
boxplot = df1.boxplot(column=['smr'])
```



Слика: 11

3.1.4. Варијанса

Варијансата е вредност која индицира како вредностите за дадена варијабла се распоредени околу аритметичката средина. Висока варијанса значи дека вредностите се на поголемо растојание, ниска варијанса значи дека се блиску густо распоредени. Оваа мерка е доста значајна кога се аплицира врз точноста на модел за предикција.

$$s^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n - 1}$$

Варијансата е секогаш позитивен број.

Варијансата за ратата на самоубиства на SHG множеството пресметано со n-1 степени на слобода:

68.24158858780554

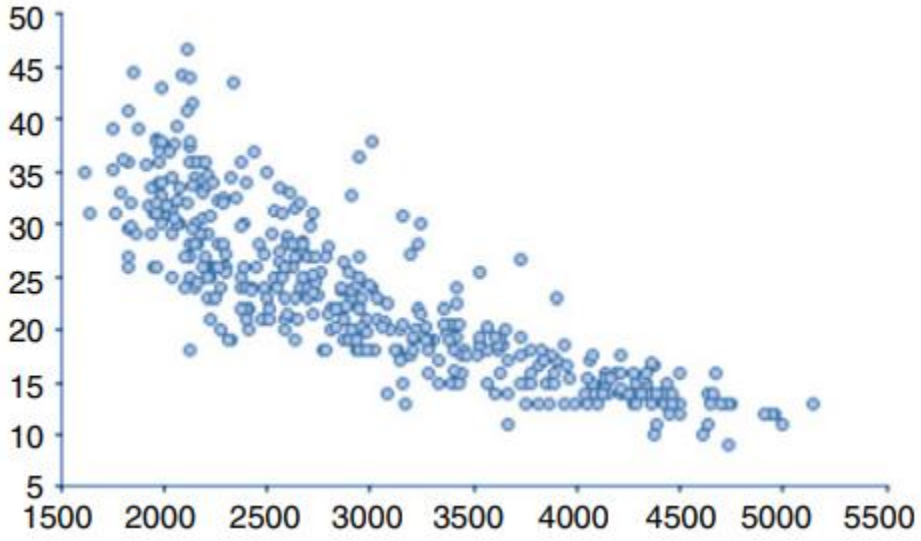
3.1.5. Стандардна девијација

Стандардната девијација е квадратен корен од варијансата. Индикатор за дисперзија исто како варијансата со тоа што мерката во која се изразува е иста со мерната единица на самата варијабла. Затоа и е почесто користена мерка од варијансата.

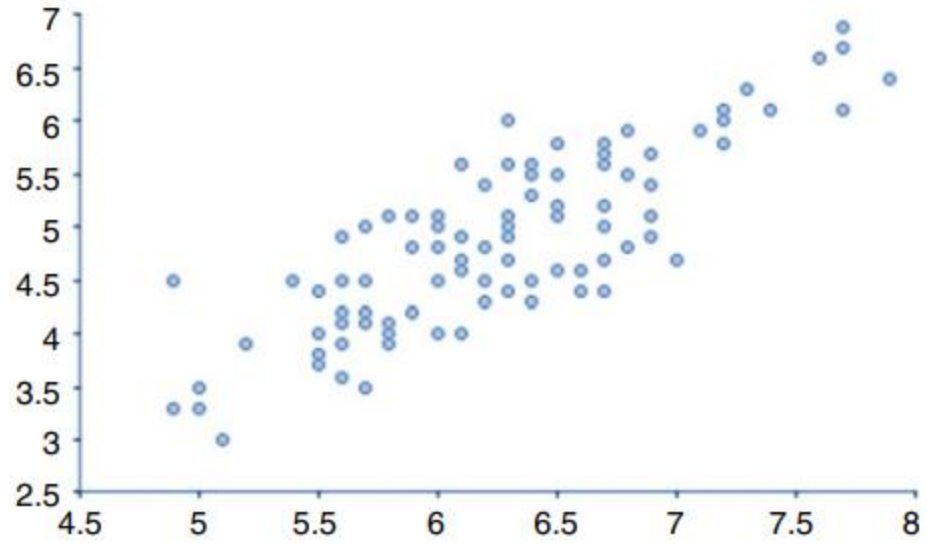
За ратата на самоубиства според SHG множеството, девијацијата е: 8.260846723417977

Корелации помеѓу варијабли

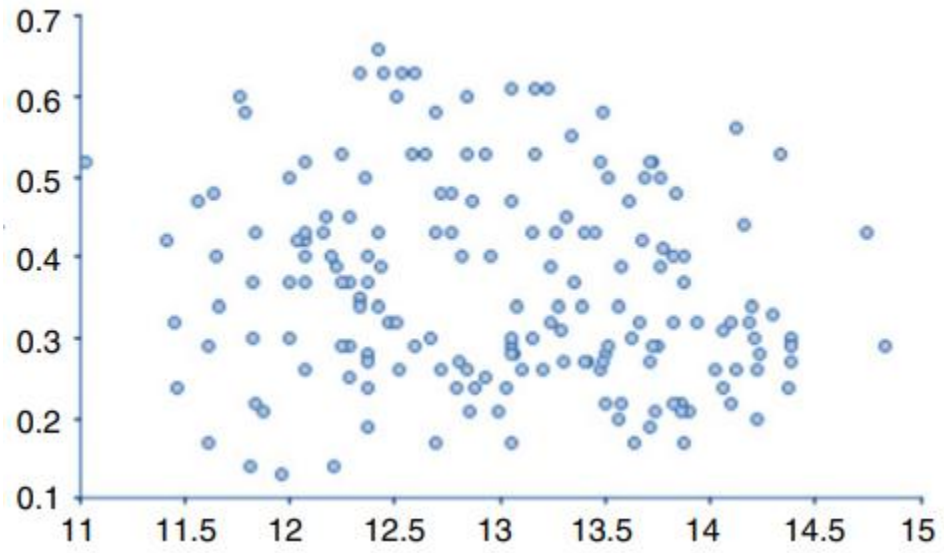
При анализа на податочно множество, не ретко има потреба од идентификација на корелација помеѓу варијаблите. За визуелизација на односот помеѓу вредностите на две варијабли најчесто се користи дијаграм на разнесување. Вредностите на едната варијабла се прикажуваат на x а од втората на y оската. Зависноста може да биде линеарно растечка или опаѓачка доколку варијаблите зависат право пропорционално или обратно пропорционално една од друга. Лесно може да се примети и нелинеарна опаѓачка или растечка зависност. Доколку варијаблите немаат корелација, точките ќе бидат расфрлани низ графикот без видлива шема. На слика 12, 13 и 14 се дадени неколку вакви примери.



Слика: 12



Слика: 13



Слика: 14

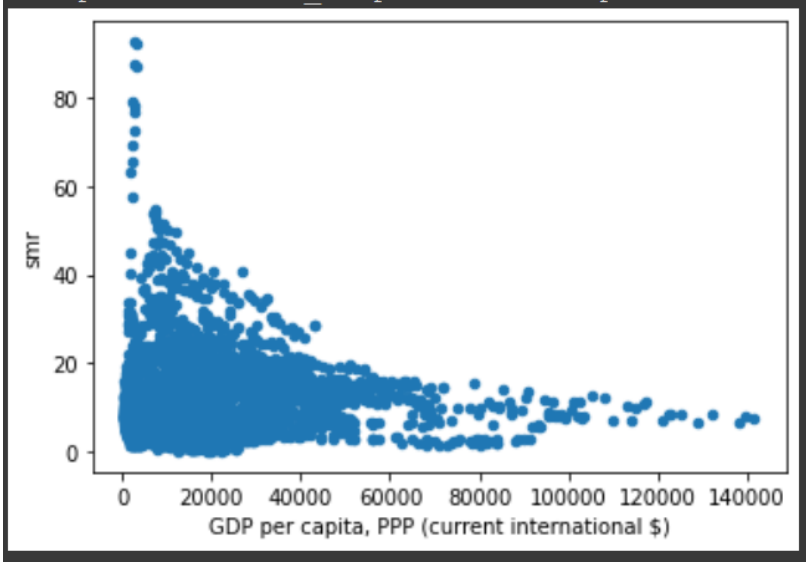
Практичен пример:

На слика 15 е даден дијаграм на расејување на број на самоубиства и висина на приход. Според дијаграмот може да се забележи извесна корелација од опаѓачка природа.

На слика 16 е даден дијаграм на расејување на број на самоубиства по година. Генерално не може да се констатира јасна зависност помеѓу променливите.

```
df1.plot.scatter("GDP per capita, PPP (current international $)","smr")
```

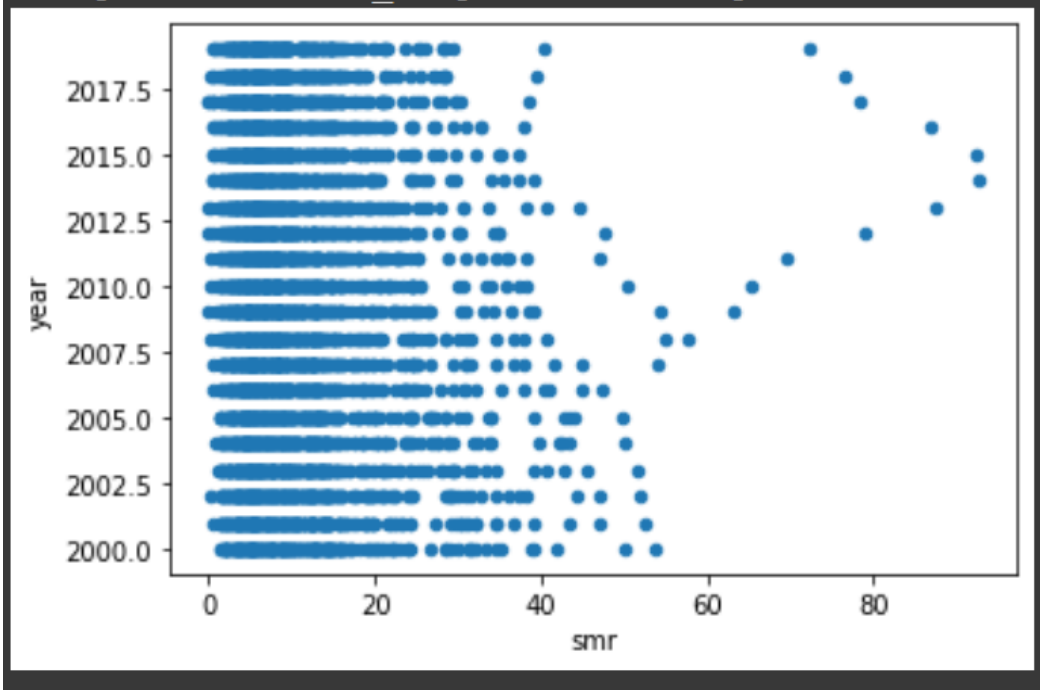
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4bfa41b850>
```



Слика: 15

```
df1.plot.scatter("smr","year")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4bfa8df4d0>
```



Слика: 16

3.1.6. Кендал Тау

Кендал Тау е еден пристап за мерење на корелација помеѓу варијиаблите со тоа што го споредува правецот на промена на вредностите, односно доколку разликата помеѓу два реда вредности за за двете варијабли е позитивна, тогаш коефициентот е позитивен. Доколку разликата е позитивна за едната а негативна за другата варијабла, коефициентот е негативен.

На слика 17 е дадена табела од корелациските коефициенти помеѓу секоја од варијаблите од SHG множеството. Корелацијата помеѓу годината и бројот на самоубиства е многу блиску до 0 (-0.055834) што значи дека варијаблите се скоро независни. Целосната зависност на варијаблата од самата себе е индицирана со вредност 1. Коефициентот помеѓу висината на приходот и ратата на самоубиства е 0.124, што значи дека постои слаба поврзаност помеѓу варијаблите.

```
df1.corr(method='kendall')
```

	year	inth	smr	GDP (current US$)	GDP per capita, PPP (current international $)
year	1.000000	-0.049441	-0.055834	0.102480	0.142997
inth	-0.049441	1.000000	-0.086019	-0.224687	-0.330088
smr	-0.055834	-0.086019	1.000000	0.092545	0.124464
GDP (current US$)	0.102480	-0.224687	0.092545	1.000000	0.304179
GDP per capita, PPP (current international $)	0.142997	-0.330088	0.124464	0.304179	1.000000

Слика: 17

4. Дефиниција на модел

Алгоритми кои може да се применат за развој на модел има голем број веќе дефинирани. Конкретната имплементација може да биде различна, но најчесто се користат готови библиотеки за да може корисникот да се фокусира на решавањето на бизнис проблемот и што помалку да се грижи за имплементациски детаљи. Такви библиотеки кои ќе бидат искористени е Scikit-learn[10] и Numpy[13].

Ќе наведеме неколку добро познати алгоритми и ќе испробаме еден од нив на демо податочно множество.

4.1. Кластерирање

Во случаеви кога работиме со податоци без лабели, развојот на модели се базира на учење без надзор. За споредба податочно множество кое со состои од слики од животни и соодветните имиња е лабелирано множество. Множество само од слики од животни е нелабелирано.

Еден од методите на учење без надзор е кластеризација. Со оваа техника, моделот се обидува да ги групира податоците во кластери преку пресметка на растојанието помеѓу самите податоци. Постојат голем број на методи на кластерирање зависно од повеќе фактори, како на пример дефиницијата за растојание. Но главниот принцип на кластерирањето е максимизацијата на сличностите помеѓу елементите во кластерот и минимизација на сличности меѓу елементите од различни кластери. Методите на кластерирање може да се поделат во 4 групи:

Кластерирање со центроиди

Хиерархиско кластерирање

Кластерирање според распределба

Кластерирање според густина

Практичната примена на кластерирањето е доста широка. Неколку примери се филтрирање на спам електронски пораки, класификација на документи, репрезентација на вистински бои во 24бит/пиксели, превод на ракописен текст.

4.1.1. Мерки за кластерирање

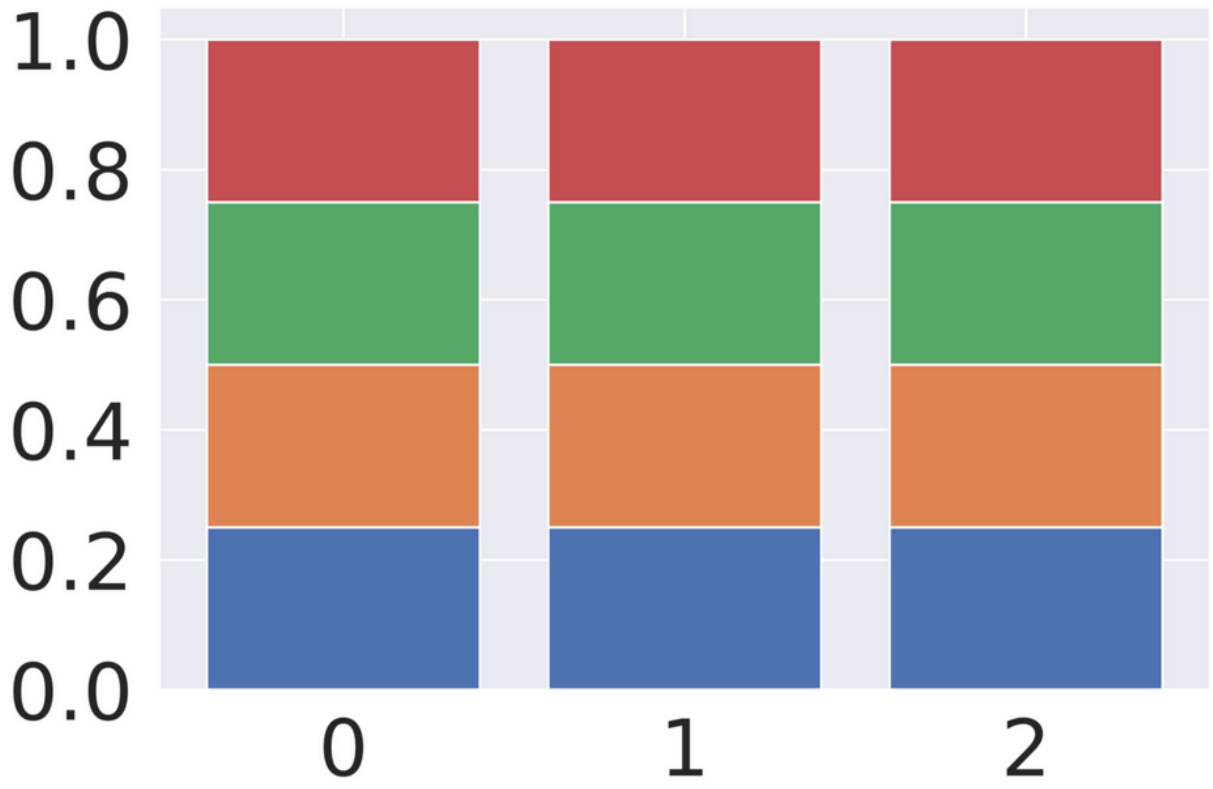
Имплементацијата на моделот на кластерирање е дел од проблемот. Втор сегмент кој што е важен е евалуација на ефикасноста и точноста на моделот. Има повеќе мерки кои може да се искористат во доменот на моделите за кластеризација. За евалуација на демо моделите ќе искористиме неколку од нив.

Хомогеност: Коефициентот на хомогеност е мерка која одредува дали точките од еден кластер припаѓаат на една иста класа, односно е мерка за сличноста помеѓу точките од еден кластер. Вредноста е помеѓу 0 и 1, со тоа што вредноста од 1 значи дека сите точки во кластерот ја имаат истата класа, 0 означува ниска хомогеност, односно сличноста помеѓу точките во кластерот е многу ниска.

Комплетност: Коефициент помеѓу 0 и 1 кој покажува дали кластерот ги опфаќа сите блиски точки, односно дали сите точки кои припаѓаат на една класа припаѓаат и на еден кластер.

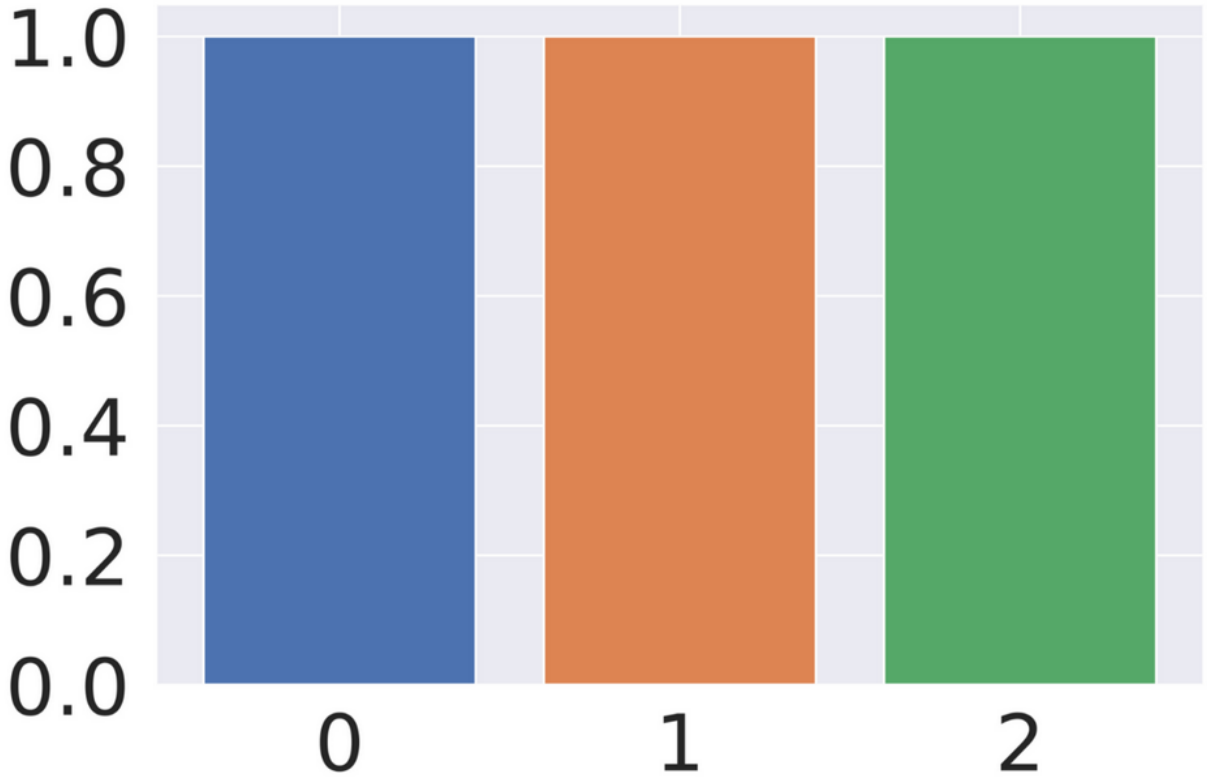
B-мерка: коефициент кој ја обединува комплетноста и хомогеноста. Вредноста може да е помеѓу 0 и 1, со тоа што вредност од 1 значи дека кластерите се сосема хомогени и комплетни, а 0 ниту хомогени ниту комплетни.

На слика 18 е даден пример за лошо кластерирање, односно модел со ниска хомогеност и комплетност. Сини точки има во секој од кластерите, значи некомплетност. Во еден кластер има точки од 4 различни бои, значи нехомогеност.



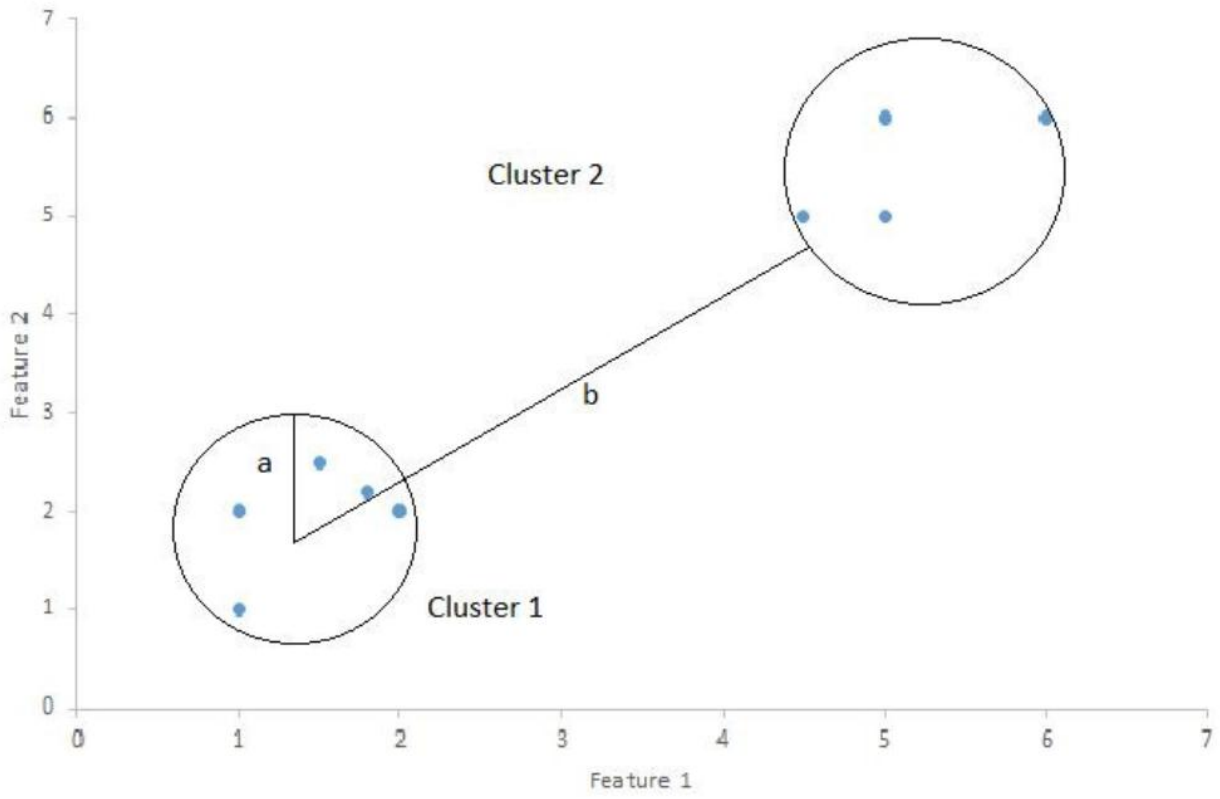
Слика: 18

Оптималното кластерирање спрема овие мерки е дадено на слика 19. Точки само од една боја има во секој од кластерите, значи висока хомогеност. Точки од една боја има само во еден кластер, значи моделот има висока комплетност.



Слика: 19

Силуетен коефициент: Земајќи го во предвид графикот на слика 20. силуетниот коефициент се пресметува како (b-a)/max(a,b) каде што a е просечното растојание помеѓу секоја точка и центроидот на соодветниот кластер, додека b е просечното интра кластерно растојание.



Слика: 20

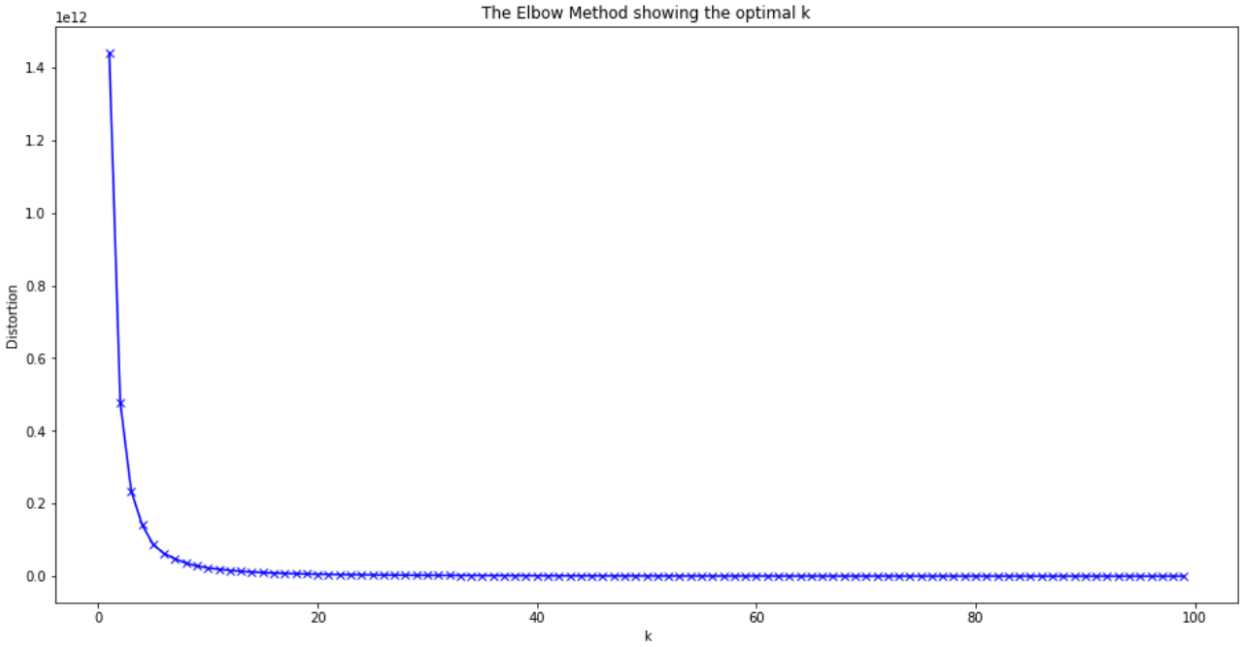
Вредноста на силуетниот коефициент е помеѓу -1 и 1, каде што вредноста 1 значи дека кластерите се добро дефинирани и оддалечени меѓусебно. Вредност 0 значи дека нема јасно дефинирани кластери, додека вредност -1 значи дека кластерите се погрешно дефинирани.

4.1.2. K-means кластерирање

Најпознат метод на кластерирање е k-means односно кластерирање со k-средини. Ова е метод на кластерирање со центроиди. Алгоритмот функционира на тој начин што одбира k произволни точки и секоја точка се алоцира на тој кластер чиј центроид е најблиску. Откако се поставени иницијалните кластери, се пресметува аритметичката средина на тој кластер и центроидот се поместува во таа точка. Бројот на итерации, односно епохи, е еден од хипер параметрите на алогиритмот кој теба да се оптимизира. Мерка за ефикасност на алгоритмот, односно колку точно е кластеризацијата, е инерцијата на моделот. Инерцијата се пресметува како сумата од квадратите на растојанијата помеѓу секоја точка и центроидот на кластерот на кој таа припаѓа. Колку е инерцијата помала, толку поточен е моделот. Инерцијата е 0 доколку, во најлош случај, бројот на кластери е еднаков со бројот на точки. Инерцијата е најголема за k=1. Затоа бројот на кластери k е еден од хипер параметрите кој подложи на оптимизација. Преку исцртување на графикот на инерцијата може да се одреди оптималниот број на кластери. Обично тој има форма на лак, каде што инерцијата брзо опаѓа а потоа опаѓа поблаго.

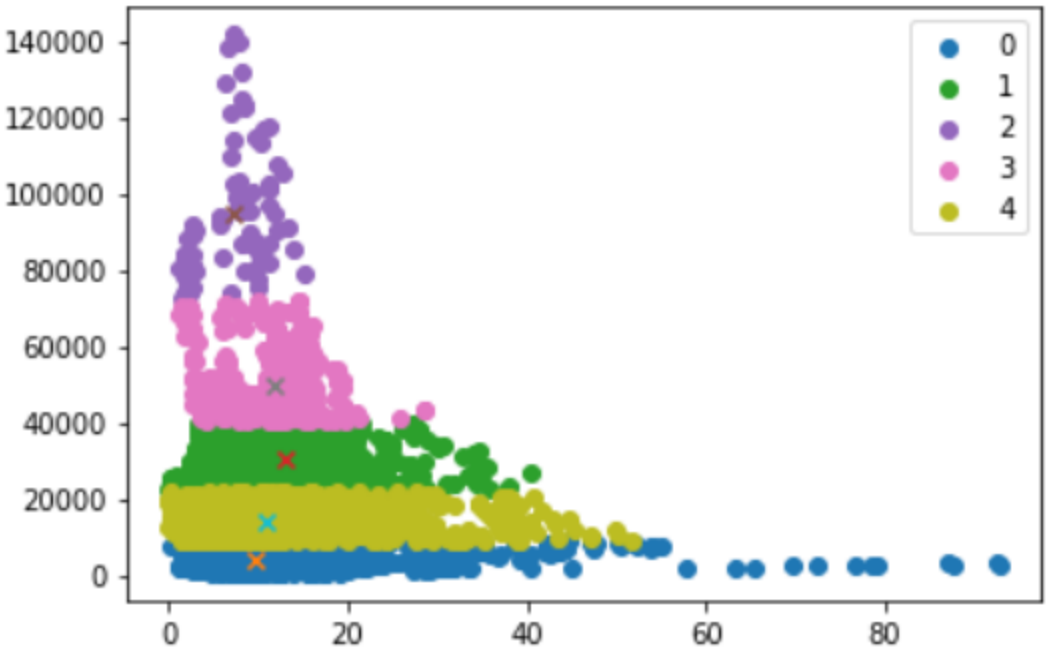
Најлесно за демонстрација се дво димензионални податоци. Како пример може да се земат колоните за рата на самоубиства и приход на државата. На слика 21. е даден графикот на инерција (или дисторзија) за k-means кластеризација на овие две колони од SHG податочното множество за

1 < k < 100 и може да се забележи дека за k > 10 веќе кривата станува скоро хоризонтална и ќе ја допре x оската во k = n, ако n е бројот на точки во податочното множество. Ова не води да размислуваме во насока на развој на модел со 3 до 5 кластери како најоптимално решение.



Слика: 21

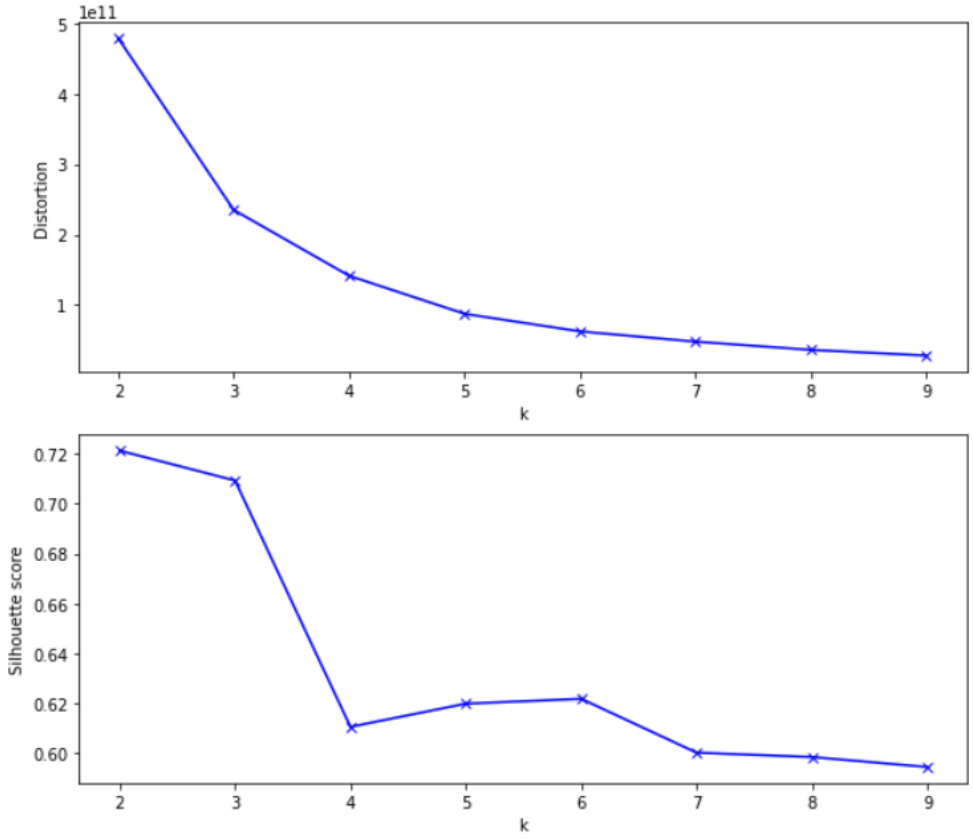
На слика 22 е даден графикот на резултатот од k-means кластеризацијата на дводимензионалните точки од ратата на самоубиства и државниот приход од SHG податочното множество за 5 центроидно кластерирање. Точките се податоците обоени според кластерот на кои припаѓаат, а со “x” се маркирани центроидите на соодветните кластери.



Слика: 22

Во принцип, центроидите иницијално се поставуваат во прозиволни точки, што во некои случаи може да предизвика конвергирање во субоптимална точност. Подобрување на алгоритмот може да се примени во иницијализацијата на центроидите. Kmeans++ алгоритмот го алоцира првиот центроид произволно и потоа го пресметува расојанието од него до секоја точка. Колку точките се подалеку, толку поголема е веројатноста дека некоја од нив ќе биде одбрана за следен центроид. Алгоритмот се повторува се додека не се алоцира предефинираниот број на центроиди.

На слика 23 се дадени еден до друг графиците на инерцијата и силуетниот коефициент во зависност од бојот на кластери k за $2 < k < 10$.



Слика: 23

4.1.3. Хиерархиско кластерирање

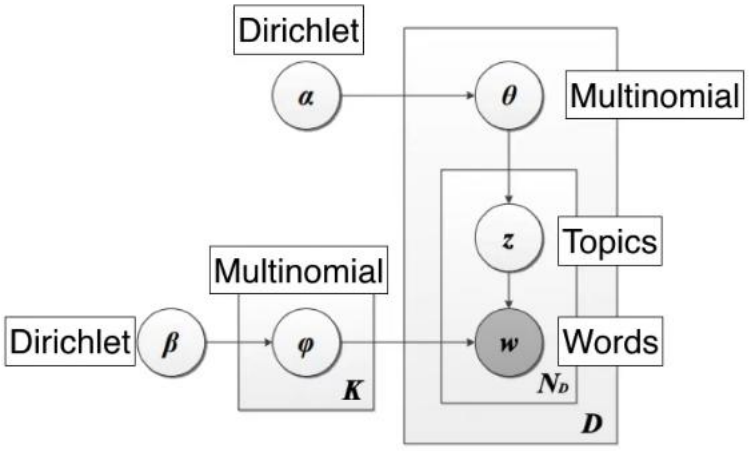
Кластерирање базирано на конективност. Алгоритмот[41] работи под претпоставка дека точките кои се поблиску се поповрзани отколку точките кои се подалеку една од друга. Тука не постои концепт на центроиди или средини. Со хиерархиско кластерирање се гради дрво структура во која листовите се точките. Предностите на хиерархиското кластерирање во однос на kmeans е тоа што може лесно да се аплицира на секакви податочни типови, дури и текст, но од друга страна пак е поскапо во однос на пресметковна моќ и може да не е најдобриот пристап доколку имаме преголемо податочно множество за обработка.

4.1.4. LDA

Latent Dirichlet Allocation е модел за веројатност кој се користи за екстракција на теми и алокација на документи по тема. LDA[18] е еден начин за решавање на класификациски проблем и спаѓа во

класата на машинско учење без надзор. Споредено со k-means класификација со LDA дозволува алокација на еден документ во повеќе теми, што е доста поприроден начин на класифицирање кога зборуваме за текст документи. На слика 24 е прикажана архитектурата на LDA. Детаљите околу функционирањето на алгоритмот не се во фокусот, бидејќи не се тема на истражување. Во главно алгоритмот работи на принцип на аплицирање на дирихле распределба на документите од корпусот по теми и дирихле распределба на темите по зборови.

$$P(\boldsymbol{W}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{\varphi}; \alpha, \beta) = \prod_{j=1}^{M} P(\theta_j; \alpha) \prod_{i=1}^{K} P(\varphi_i; \beta) \prod_{t=1}^{N} P(Z_{j,t} \mid \theta_j)\, P(W_{j,t} \mid \varphi_{Z_{j,t}})$$



Слика: 24

Примената на LDA е генерирање на тагови, генерирање на документи по тагови, класифицирање на документи. Пример практична примена: автоматско сортирање на електронско сандаче, препораки за книги, филмови, блогови или било какви други текст содржини.

4.1.4.1. Демо

За демонстрација ќе искористиме податочно множество: сите реплики од серијата „Игра на тронови“ преземено од Kaggle. Кодот го пишуваме во Python. За манипулација со податоците користиме Pandas. За манипулација со зборовите користиме NLTK[43], тоа е платформа изградена врз Python која меѓу другото содржи и библиотеки за класификација, токенизација, семантички и други корисни операции врз текст. Ќе ја користиме ЛДА имплементацијата од Gensim[42]. ЛДА ќе го искористиме за генерирање на тагови.

Како што објаснивме на почеток, најпрво го вчитуваме податочното множество и анализираме какви податоци имаме на располагање и во кој формат (слика 25).

```
df_got = pd.read_csv(df)
# check first 5 records
df_got.head(n=5)
```

	Release Date	Season	Episode	Episode Title	Name	Sentence
0	2011-04-17	Season 1	Episode 1	Winter is Coming	waymar royce	What do you expect? They're savages. One lot s...
1	2011-04-17	Season 1	Episode 1	Winter is Coming	will	I've never seen wildlings do a thing like this...
2	2011-04-17	Season 1	Episode 1	Winter is Coming	waymar royce	How close did you get?
3	2011-04-17	Season 1	Episode 1	Winter is Coming	will	Close as any man would.
4	2011-04-17	Season 1	Episode 1	Winter is Coming	gared	We should head back to the wall.

Слика: 25

Следен чекор е чистење на податоците. За таа цел ја користиме NLTK библиотеката. Прво ја импортираме и потоа ги иницијализираме објектите и статичките листи од токени кои треба да се отстранат. Отстрануваме интерпункција. Дополнително, со анализа на податоците забележано е дека многу од заменките и глаголите не се правилно напишани, односно theyre наместо they’re, што значи дека бибилиотеката нема автоматски да ги препознае и отстрани, затоа ги додаваме на листата за отстранување. На истата листа додаваме и уште неколку зборови кои се пречести и ќе ги искриват резултатите. Слика 26.

```
import nltk
from nltk.tokenize import word_tokenize
from collections import Counter
nltk.download('wordnet')      #download if using this module for the first time

from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
nltk.download('stopwords')    #download if using this module for the first time
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
#For Gensim
import gensim
import string
from gensim import corpora
from gensim.corpora.dictionary import Dictionary
from nltk.tokenize import word_tokenize

stopwords = set(stopwords.words('english')).union(set(["man","men","dont", "theyre","were","Theyre","theyll","ha","shell","Were","Im","youre","Youre"
exclude = set("!\"#$%&()*+,-./:;<=>?@[\]^_{|}~'`")
lemma = WordNetLemmatizer()
```

Слика: 26

Следно, дефинираме функција за чистење и подесуваме дека сакаме да задржиме само именки од целиот текст – NN означува именки, а NNS се именки во множина. Слика 27.

```python
def clean(document):
    document = str(document)
    words = document.split(' ')
    words = " ".join(lemma.lemmatize(word) for word in words)
    words = ''.join(ch for ch in words if ch not in exclude).split(' ')
    words = [i for i in words if i not in stopwords]
    words = ' '.join(words)
    tokens = nltk.word_tokenize(words)
    tags = nltk.pos_tag(tokens)
    nouns = [word.lower() for word,pos in tags if (pos == 'NN' or pos == 'NNS')]

    return nouns
```

Слика: 27

Пуштаме текстот да се исчисти и проверуваме како сега изгледа податочното множество (слика 28).

```
clean_episodes.head()

0    [lot, steal, goat, lot, pieces]
1    [wildlings, thing, thing, life]
2                              [get]
3                                 []
4                             [wall]
Name: Sentence, dtype: object
```

Слика: 28

Алгоритмот се чини работи очекувано: множеството е сведено на именки. Следно, може да го пуштиме множеството низ ЛДА алгоритмот. Подесувањата се претежно претсетирани, само повикуваме генерирање на 5 теми, со по 5 зборови. Пуштаме два пати и ги анализираме резултатите (слика 29 и 30).

```
result = lda_tags(clean_episodes)

#topic:
family
war
home
queen
moment
#topic:
theres
word
whats
end
part
#topic:
brother
dwarf
uncle
lead
houses
#topic:
place
years
everything
doesnt
walk
#topic:
face
story
soldiers
wouldnt
problem
```

Слика: 29

```
result2 = lda_tags(clean_episodes

#topic:
head
doe
dog
smile
traitors
#topic:
help
winter
position
mistake
prince
#topic:
mother
house
bit
battle
mothers
#topic:
lady
idea
fuck
wait
bloody
#topic:
thing
home
year
force
pay
```

Слика: 30

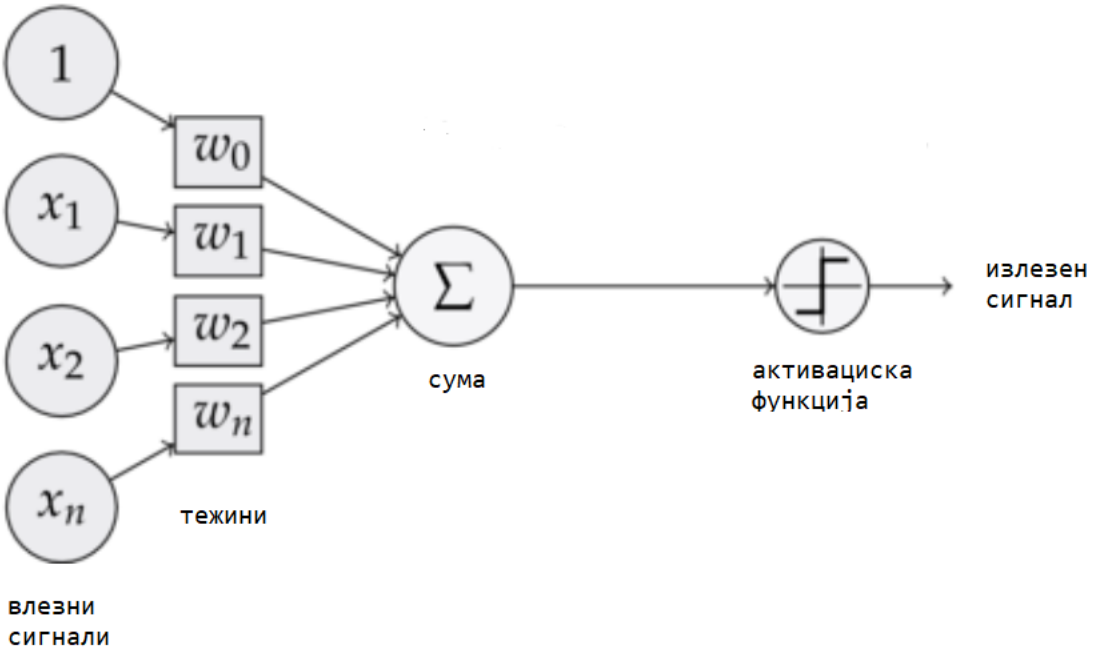
Не сите тагови се релевантни, но некои се доста репрезентативни за содржината на филмот. Кралица, војна, џуџе, фамилија, принц, зима, предавници – овие даваат релативно точна информација за што се работи во филмот, но „не би“, место, воини, лице, проблем – се малку премногу генерални за да бидат корисни. Сепак, многу лесно добиваме некаква појдовна точка, информација која може да биде корисна и процес кој може да се подобри и автоматизира.

4.2. Невронски мрежи

Невронските мрежи[19] се подмножество на машинското учење и се клучниот елемент во областа на длабоко учење. Невронските мрежи се множество алгоритми моделирани слично на човечкиот мозок, дизајнирани со цел препознавање на шаблони. Шаблоните кои можат да бидат препознаени се нумерички во форма на вектори. При решавање на даден проблем, податоците кои треба да бидат обработени, без разлика дали се слика, текст, звук или временски серии, мора

прво да бидат претворени во формат на нумерички вектори за да може да бидат обработени од невронскиот модел.

Во основа, една невронска мрежа е составена од јазли кои се аналогни на невроните во човечкиот мозок и секој јазол е поврзан со еден или повеќе други јазли преку врски аналогни на мозочните синапси. На слика 31 е даден математички модел на неврон. Од сликата се гледа дека секој неврон се состои од дефинирани тежини и активациска функција. Врските помеѓу невроните ги носат влезните параметри до невронот. Секоја од овие врски има извесна тежина. На секоја од влезните вредности се аплицира соодветната тежина и добиените вредности се собираат. Сумата се пушта како влезен параметар на активациската функција. Оваа функција го дефинира прагот на активација на невронот, односно ако вредноста е под тој праг, невронот нема да пропагира вредност кон следниот неврон во мрежата. Во зависност од тоа каква функција се користи тука, се пресметува излезниот сигнал од јазелот.



Слика: 31

Кога зборуваме за активациска функција, не постои ограничување за истата. Може да се користи било каква функција, но обично се практикува употреба на нелинеарни функции бидејќи така се овозможува моделирање на покомплексни одлучувања.

4.2.1. Активациски функции

4.2.1.1. Функција на бинарен чекор

Функцијата на бинарен чекор зависи од прагот што одлучува дали невронот треба да се активира или не. Влезниот параметар што се внесува во функцијата за активирање се споредува со одреден

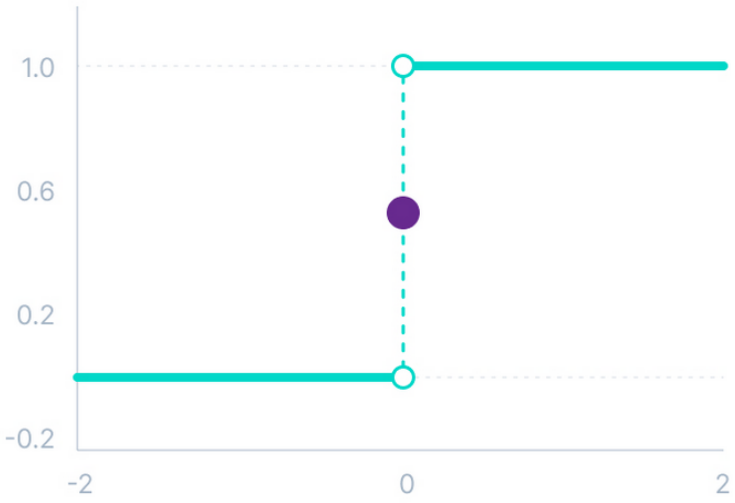
праг; ако влезот е поголем од него, тогаш невронот се активира, во спротивно се деактивира, што значи дека неговиот излез не се пренесува на следниот неврон.

Ограничувањата на функцијата бинарен чекор:

Не може да обезбеди излез со повеќе вредности - на пример, не може да се користи за проблеми со класификација со повеќе класи.

Градиентот на функцијата е нула, што го попречува процесот на повратна пропагација. На слика 32 е даден графикот на функцијата.

$$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geqslant 0 \end{cases}$$

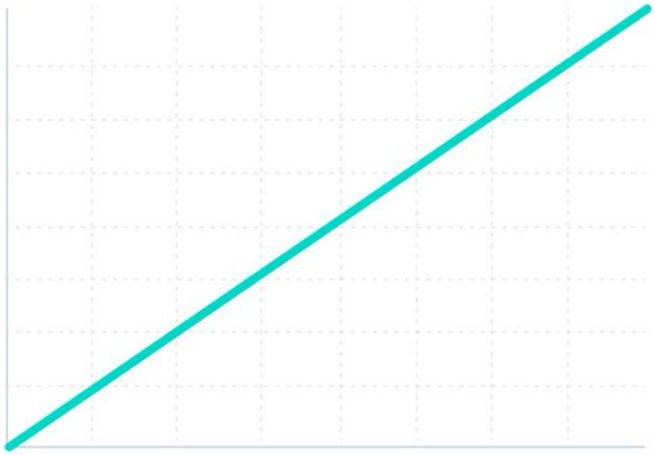


Слика: 32

4.2.1.2. Линеарна активациска функција

Линеарната активациска функција е едноставна праволиниска функција која е директно пропорционална со влезот. Ја има равенката:

$$f(x) = kx$$



Слика: 33

На слика 33 е даден графикот на функцијата.

За разлика од функцијата со бинарен чекор, функцијата за линеарно активирање може да се справи со повеќе од една класа како излез, но има свои недостатоци.

Проблемот со линеарната функција за активирање е што не може да се дефинира во одреден опсег. Има опсег од (-∞, ∞). Без разлика колку слоеви има невронската мрежа, последниот слој секогаш работи како линеарна функција на првиот слој. Ова ја прави невронската мрежа неспособна да се справи со сложените проблеми.

Друг проблем е што градиентот е константа и воопшто не зависи од влезот. Како резултат на тоа за време на заднинското ширење стапката на промена на грешката е константна. Така невронската мрежа навистина нема да се подобри со постојан градиент.

4.2.1.3. Нелинеарни функции за активирање

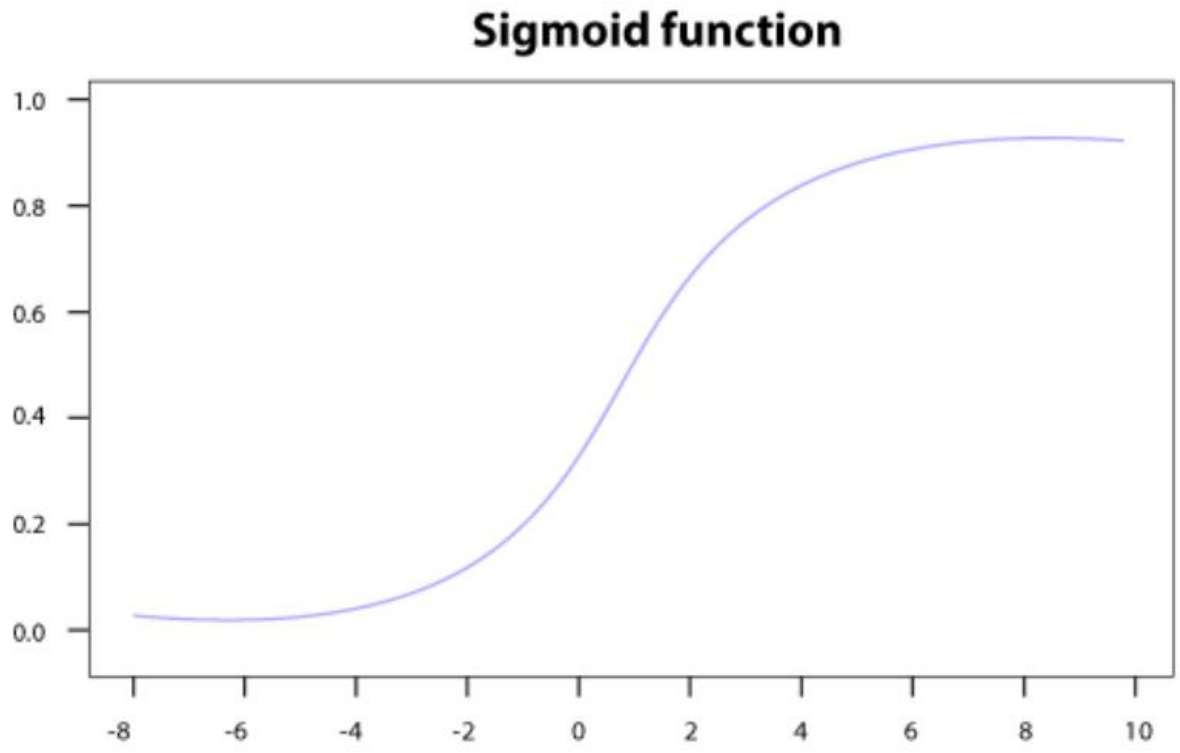
Функцијата за линеарно активирање прикажана погоре е едноставно модел на линеарна регресија. Поради неговата ограничена моќност, ова не му дозволува на моделот да создава сложени мапирања помеѓу влезовите и излезите на мрежата. Нелинеарните функции за активирање ги решаваат следните ограничувања на функциите за линеарно активирање:

- Дозволуваат пропагација кон назад бидејќи сега дериватната функција би била поврзана со влезот, и можно е да се вратиме назад и да разбереме кои тежини во влезните неврони можат да обезбедат подобро предвидување.
- Дозволуваат натрупување на повеќе слоеви на неврони бидејќи излезот сега би бил нелинеарна комбинација на влез кој се пренесува низ повеќе слоеви. Секој излез може да се претстави како функционална пресметка во невронска мрежа.

4.2.1.4. Сигмоидна / логистичка функција за активирање

Оваа функција ја зема секоја реална вредност како влезна вредност и ги дава вредностите во опсег од 0 до 1. На слика 34 е даден графикот на функцијата.

Колку е поголем влезот (попозитивен), толку поблиску излезната вредност ќе биде до 1,0, додека колку е помал влезот (понегативен), толку излезот ќе биде поблиску до 0,0, како што е прикажано подолу.



Слика: 34

Сигмоидна/логистичка функција за активирање

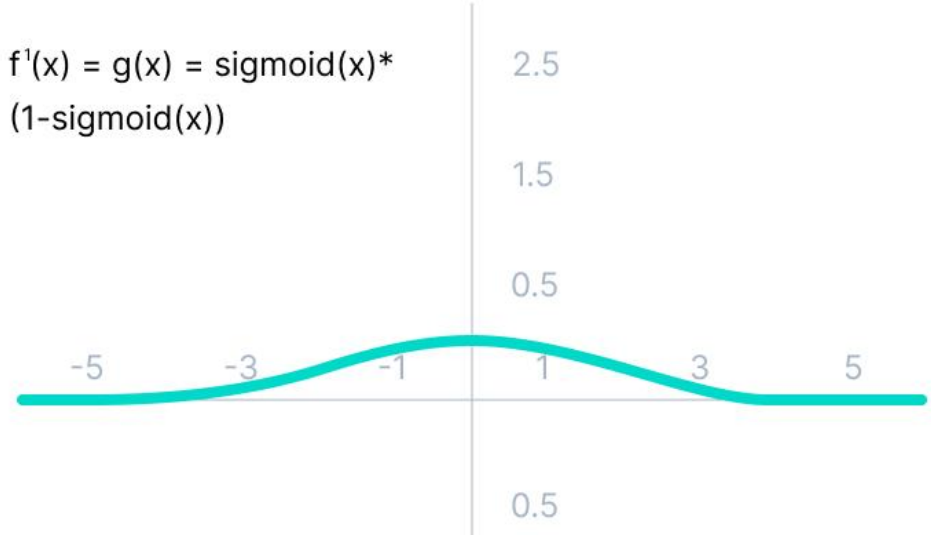
Математички може да се претстави како:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Најчесто се користи за модели каде што треба да ја предвидиме веројатноста како излез. Бидејќи веројатноста за нешто постои само помеѓу опсегот од 0 и 1, сигмоидот е најчест избор поради неговиот опсег.

Функцијата е диференцијабилна и обезбедува мазен градиент, т.е. спречување на скокови во излезните вредности. Ова е претставено со S-облик на функцијата за активирање на сигмоидот.

Ограничувањата на сигмоидната функција:



Слика: 35

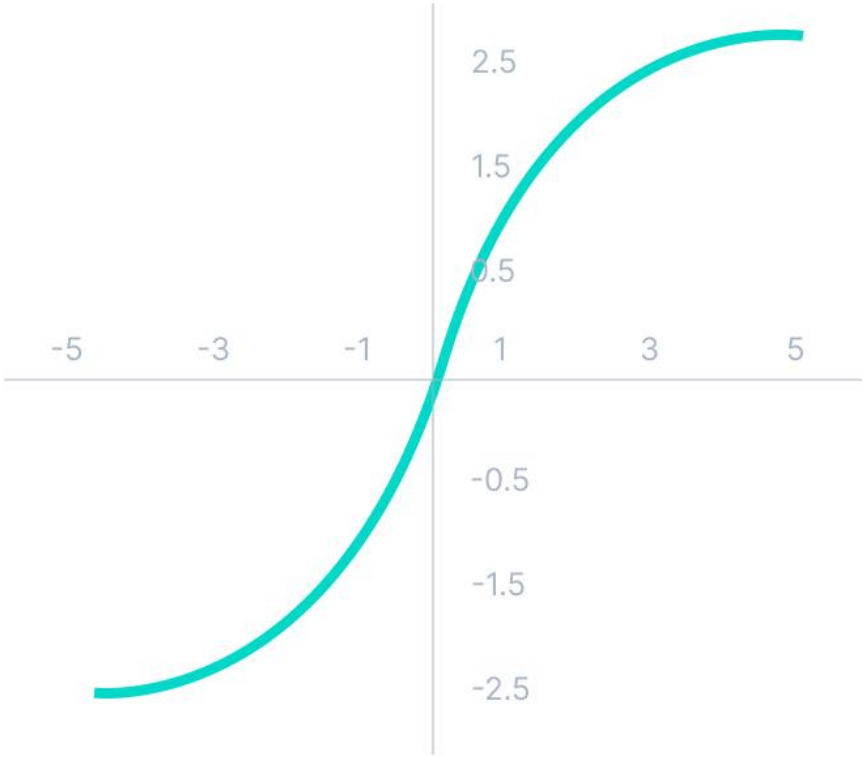
Како што можеме да видиме на слика 35, вредностите на градиентот се значајни само за опсегот -3 до 3, а графикот станува многу порамен во другите региони.

Тоа имплицира дека за вредности поголеми од 3 или помали од -3, функцијата ќе има многу мали градиенти. Како што вредноста на градиентот се приближува до нула, мрежата престанува да учи. Ова е проблем познат како исчезнувачки градиент.

Излезот на логистичката функција не е симетричен околу нула. Ова ја прави обуката на невронската мрежа потешка и нестабилна.

4.2.1.5. Tanh функција (хиперболичен тангенс)

Tanh функцијата е многу слична со функцијата за сигмоидна/логистичка активација, па дури и има ист S-облик со разлика во опсегот на излезот од -1 до 1. Во Tanh, колку е поголем влезот (попозитивен), толку е поблиску излезната вредност ќе биде до 1,0, додека колку е помал влезот (понегативен), толку излезот ќе биде поблиску до -1,0. На слика 36 е даден графикот на функцијата.



Слика: 36

Математички може да се претстави како:

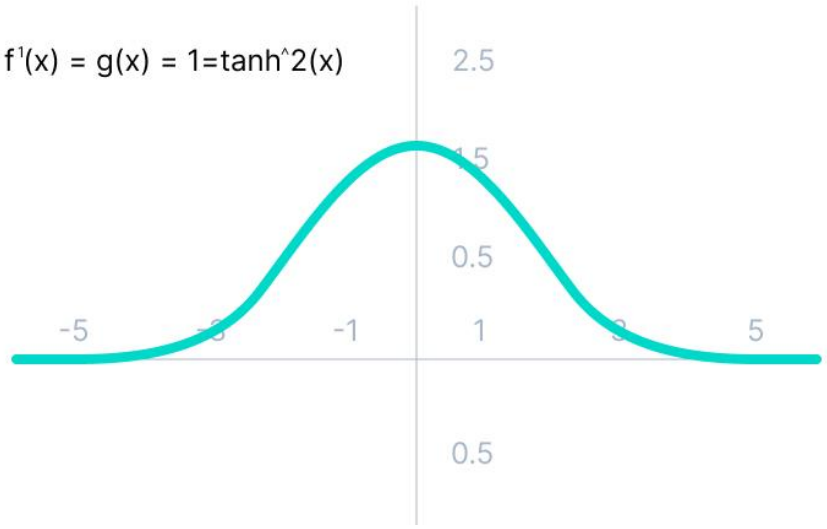
$$f(x) = \frac{\left(e^{x} - e^{-x}\right)}{\left(e^{x} + e^{-x}\right)}$$

Предностите на користењето на оваа функција за активирање се:

Излезот на функцијата за активирање tanh е центриран на нула; оттука можеме лесно да ги мапираме излезните вредности како силно негативни, неутрални или силно позитивни.

Обично се користи во скриени слоеви на невронска мрежа бидејќи нејзините вредности лежат помеѓу -1 до 1 затоа, просечната вредност за скриениот слој е 0 или многу блиску до него. Тоа помага во центрирањето на податоците и го олеснува учењето за следниот слој.

На слика 37 е даден изводот од оваа функција.



Слика: 37

Како што можете да се види тој исто така се соочува со проблемот на исчезнување на градиент слично на сигмоидната активирачка функцијла. Дополнително, градиентот на функцијата tanh е многу поостра во споредба со сигмоидна функција.

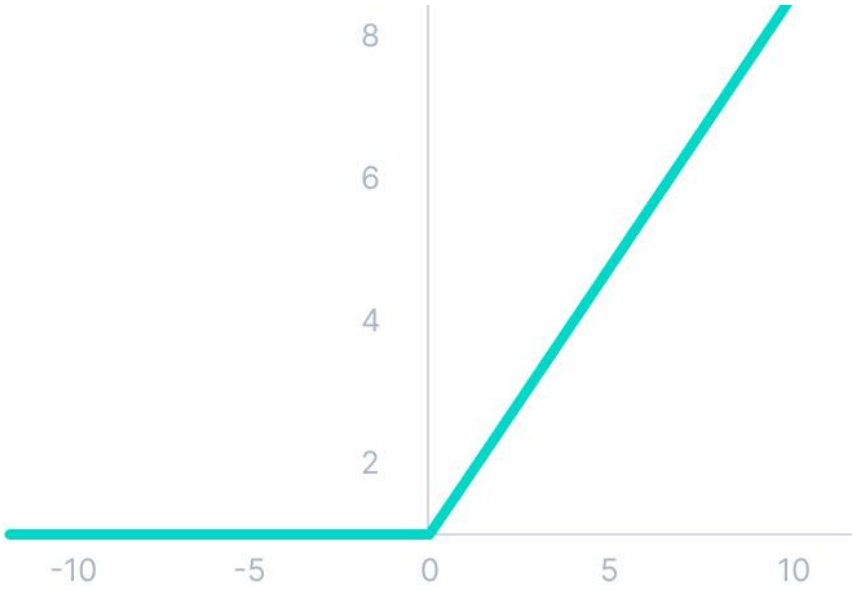
Иако и сигмоидната и tanh се соочуваат со проблем со исчезнување на градиентот, tanh е центриран на нула, а градиентите не се ограничени да се движат во одредена насока. Затоа, во пракса, tanh нелинеарноста секогаш се претпочита отколку сигмоидна нелинеарност.

4.2.1.6. Функција ReLU

ReLU[40] е кратенка за ректифицирана линеарна единица.

Иако дава впечаток на линеарна функција, ReLU има дериватива функција и овозможува пропагирање назад додека истовремено ја прави пресметковно ефикасена.

Клучен момент со оваа функција е е дека функцијата ReLU не ги активира сите неврони во исто време. Невроните ќе се деактивираат кога излезот од линеарната трансформација е помал од 0. На слика 38 е даден графикот на функцијата.



Слика: 38

Функција за активирање ReLU математички може да се претстави како:

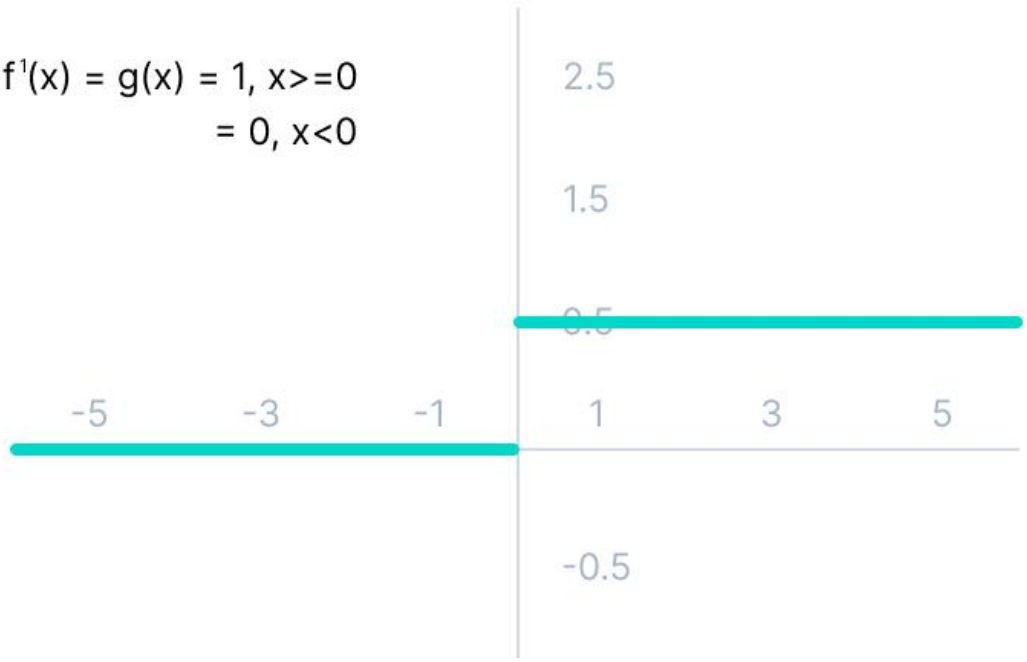
$$f(x) = max\ (0, x)$$

Предностите од користењето на ReLU како функција за активирање се како што следува:

Бидејќи само одреден број на неврони се активирани, функцијата ReLU е многу поефикасна од аспект на пресметковна моќ во споредба со сигмоидните и тан функциите.

ReLU ја забрзува конвергенцијата на спуштањето на градиент кон глобалниот минимум на функцијата на загуба поради неговото линеарно својство. На слика 39 е даден графикот на дериватот на функцијата.

Ограничувањата со кои се соочува оваа функција се:



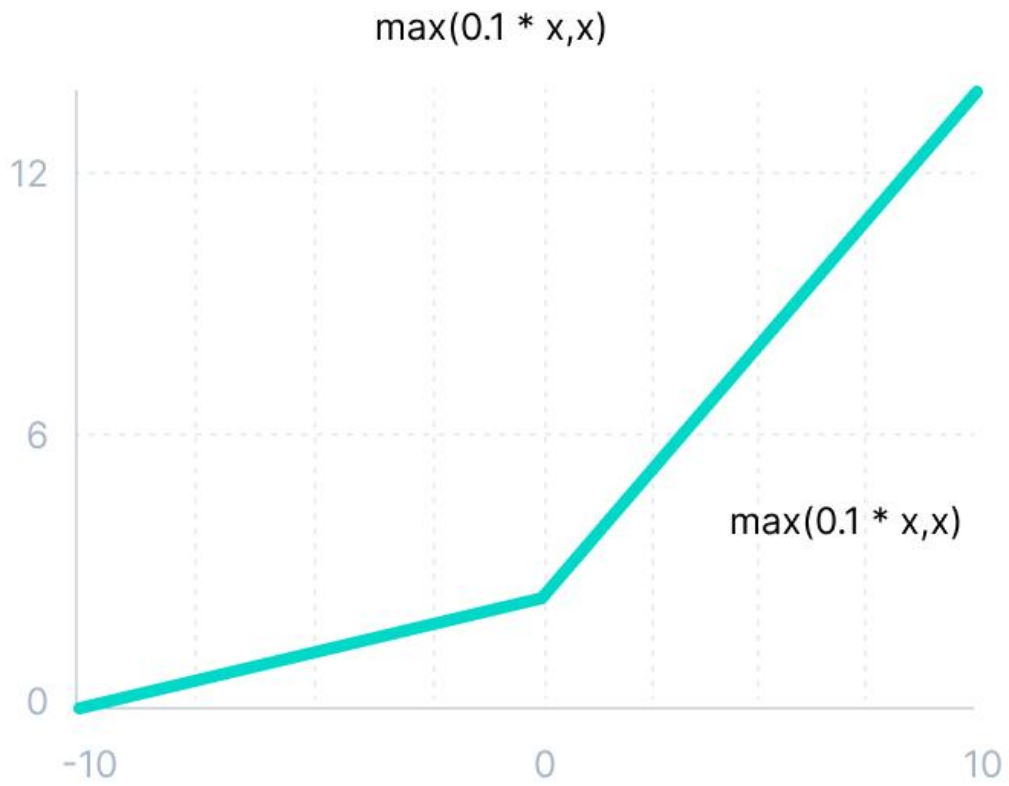
Слика: 39

Негативната страна на графикот ја прави вредноста на градиентот нула. Поради оваа причина, за време на процесот на задна пропагација, тежините за некои неврони не се ажурираат. Ова може да создаде мртви неврони кои никогаш не се активираат.

Сите негативни влезни вредности веднаш стануваат нула, што ја намалува способноста на моделот правилно да се вклопи или тренира со податоците.

4.2.1.7. Функција ReLU што протекува

Leaky ReLU е подобрена верзија на функцијата ReLU за да се реши проблемот со Dying ReLU бидејќи има мал позитивен наклон во негативната област. На слика 40 е даден графикот на функцијата.



Слика: 40

Оваа функција математички може да се претстави како:

$$f(x) = max\ (0.1x, x)$$

Предностите на Leaky ReLU се исти како оние на ReLU, покрај фактот што овозможува назад пропагирање, дури и за негативни влезни вредности.

Со правење на оваа мала модификација за негативните влезни вредности, градиентот на левата страна на графикот излегува дека е вредност која не е нула. Затоа, повеќе не би се сретнале со мртви неврони во тој регион.



Слика: 41

На слика 41 е даден првиот извод од leaky ReLU.

Ограничувањата со кои се соочува оваа функција вклучуваат:

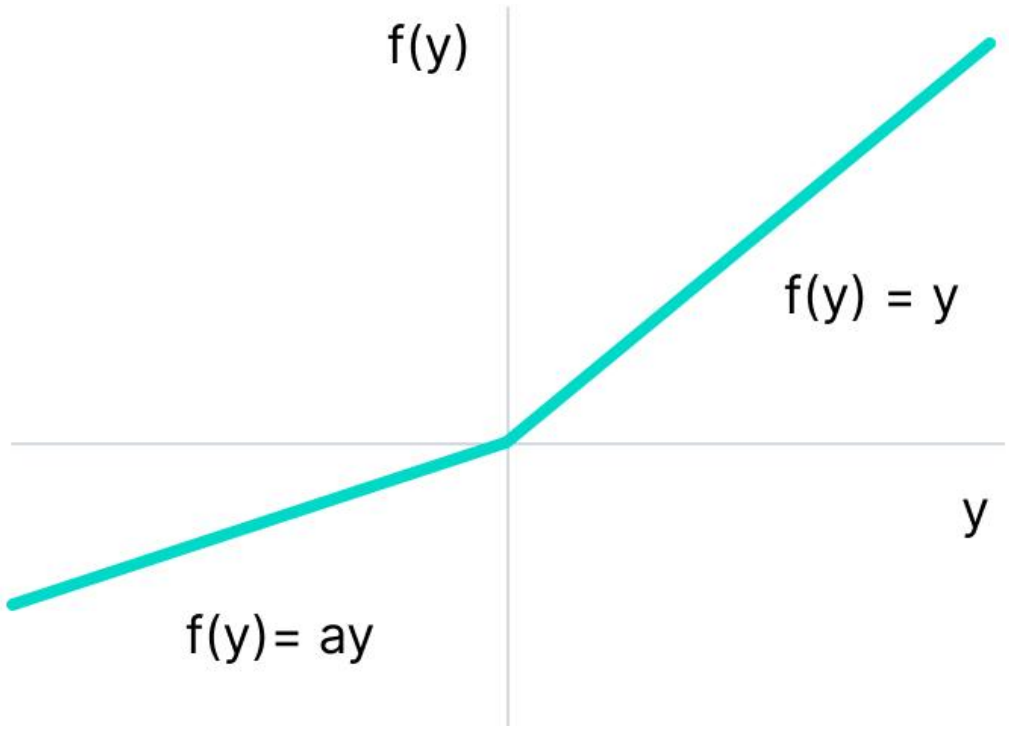
Предвидувањата може да не се конзистентни за негативни влезни вредности.

Градиентот за негативни вредности е мала вредност што предизвикува учењето на параметрите на моделот да одзема повеќе време.

4.2.1.8. Параметриска ReLU функција

Параметрискиот ReLU е друга варијанта на ReLU која има за цел да го реши проблемот дека градиентот од левата страна на оската преоѓа во нула.

Оваа функција го дава наклонот на негативниот дел од функцијата како аргумент *α*. Со извршување на пропагација кон назад се учи најсоодветната вредност на *α*. На слика 42 е даден графикот на функцијата.



Слика: 42

Параметриската ReLU функција математички може да се претстави како:

$$f(x) = max\,(ax, x)$$

Каде што *„a“* е параметарот на наклонот за негативните вредности.

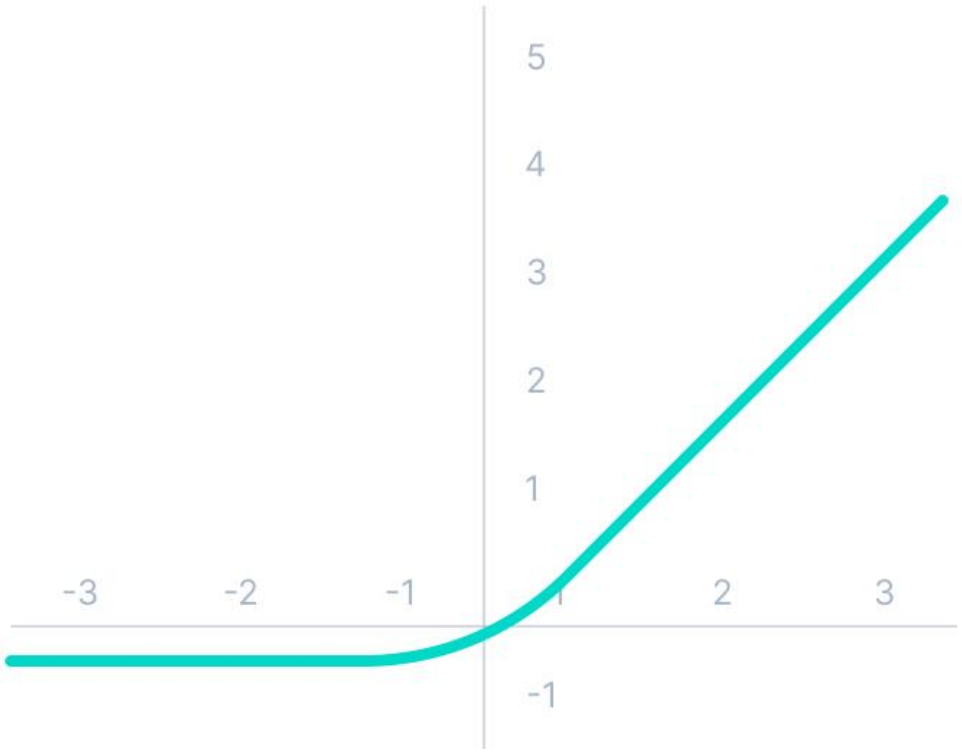
Параметризираната функција ReLU се користи кога Leaky ReLU сè уште не успева да го реши проблемот со мртвите неврони, а релевантните информации не се успешно префрлени на следниот слој.

Ограничувањето на оваа функција е тоа што таа може да работи различно за различни проблеми во зависност од вредноста на параметарот на наклон ***a.***

4.2.1.9. Функција на експоненцијални линеарни единици (ELUs)

Експоненцијална линеарна единица, или скратено ELU, е исто така варијанта на ReLU која го модифицира наклонот на негативниот дел од функцијата.

ELU користи логаритамска крива за да ги дефинира негативните вредности за разлика од Leaky ReLU и параметриска ReLU функција кои дефинираат права линија. На слика 43 е даден графикот на функцијата.



Слика: 43

Оваа функција математички може да се претстави како:

$$\begin{cases} x & for\ x \geqslant 0 \\ \alpha\left(e^{x}-1\right) & for\ x < 0 \end{cases}$$

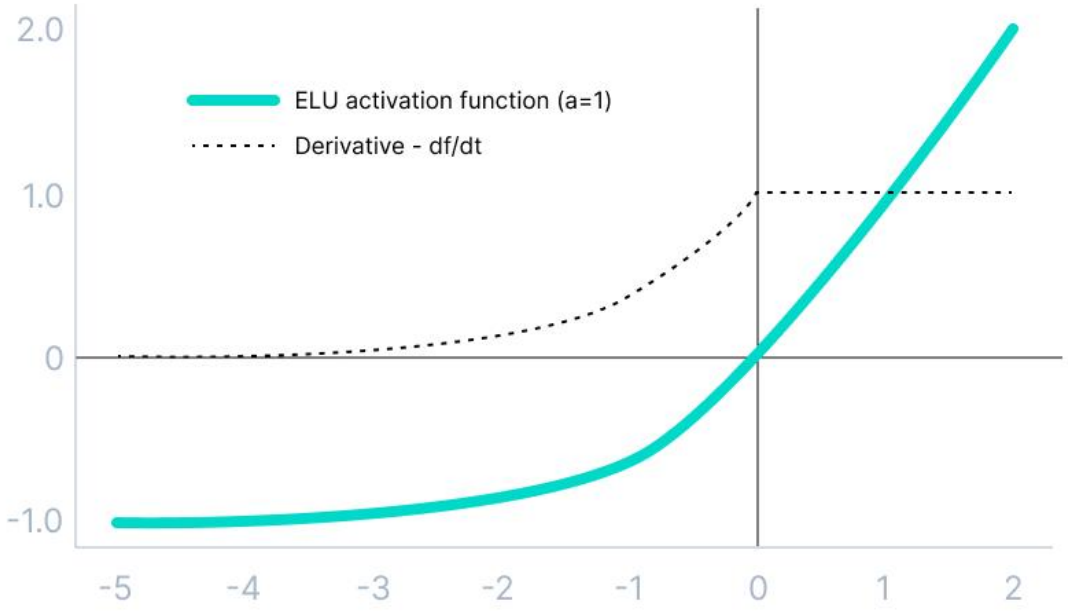
ELU е адекватна алтернатива за ReLU поради следните предности:

ELU станува рамна полека додека неговиот излез не е еднаков на –α, додека RELU нагло се израмнува.

Избегнува проблем на мртов ReLU со воведување логаритамска крива за негативни вредности на влезот. Тоа ѝ помага на мрежата да ги поттикне тежините во вистинската насока.

Ограничувањата на функцијата ELU се следните:

- Го зголемува времето на пресметување поради вклучената експоненцијална операција
- Не се учи вредноста „a“
- Проблем со експлозија на градиент



Слика: 44

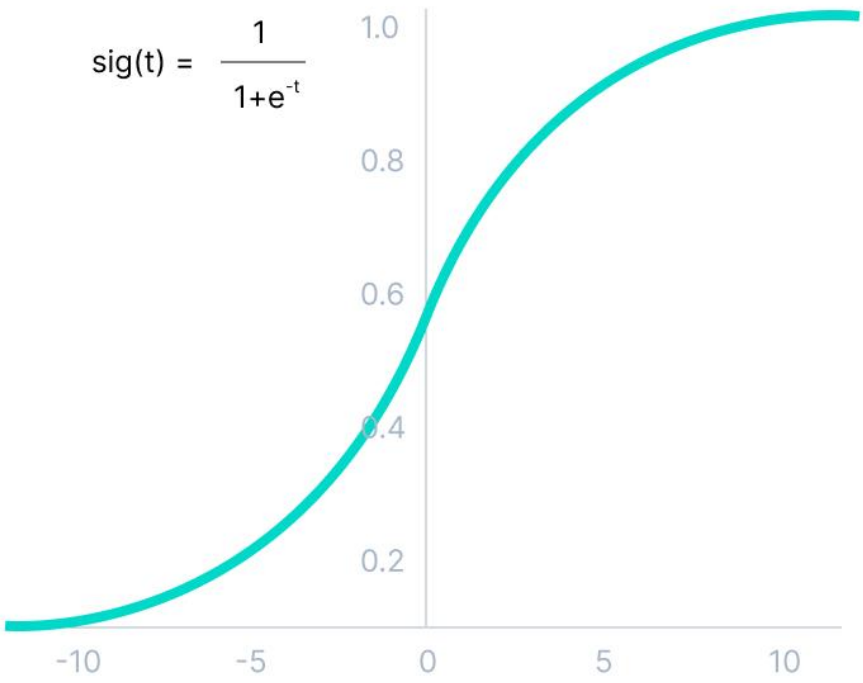
Првиот извод на оваа функција е даден на слика 44 и математички може да се претстави како:

$$f'(x) = \begin{cases} 1 & for\ x \geqslant 0 \\ f(x) + \alpha & for\ x < 0 \end{cases}$$

.

4.2.1.10. Функција Softmax

Пред да ги истражиме работите на функцијата за активирање Softmax, треба да се фокусираме на нејзиниот градежен блок - функцијата за сигмоидна/логистичка активација која работи на пресметување на вредностите на веројатност. Графикот е даден на слика 45.



Слика: 45

Излезот на сигмоидната функција е во опсег од 0 до 1, што може да се смета како веројатност.

Но, оваа функција се соочува со одредени проблеми.

Да претпоставиме дека имаме пет излезни вредности од 0,8; 0,9; 0,7; 0,8 и 0,6, соодветно. Овие излезни вредности не може да се пропагираат бидејќи немаат смисла. Односно, збирот на сите класи/излезни веројатности треба да биде еднаков на 1.

Функцијата Softmax е опишана како комбинација од повеќе сигмоиди. Ги пресметува релативните веројатности. Слично на функцијата за сигмоидна/логистичка активација, функцијата SoftMax ја враќа веројатноста за секоја класа.

Најчесто се користи како функција за активирање за последниот слој на невронската мрежа во случај на класификација со повеќе класи.

Математички може да се претстави како:

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

Ако има три класи, тоа значи дека ќе има три неврони во излезниот слој. Ако претпоставиме дека излезот од невроните е [1,8, 0,9, 0,68], примената на функцијата softmax над овие вредности за да се даде веројатен приказ ќе резултира со следниов исход: [0,58, 0,23, 0,19].

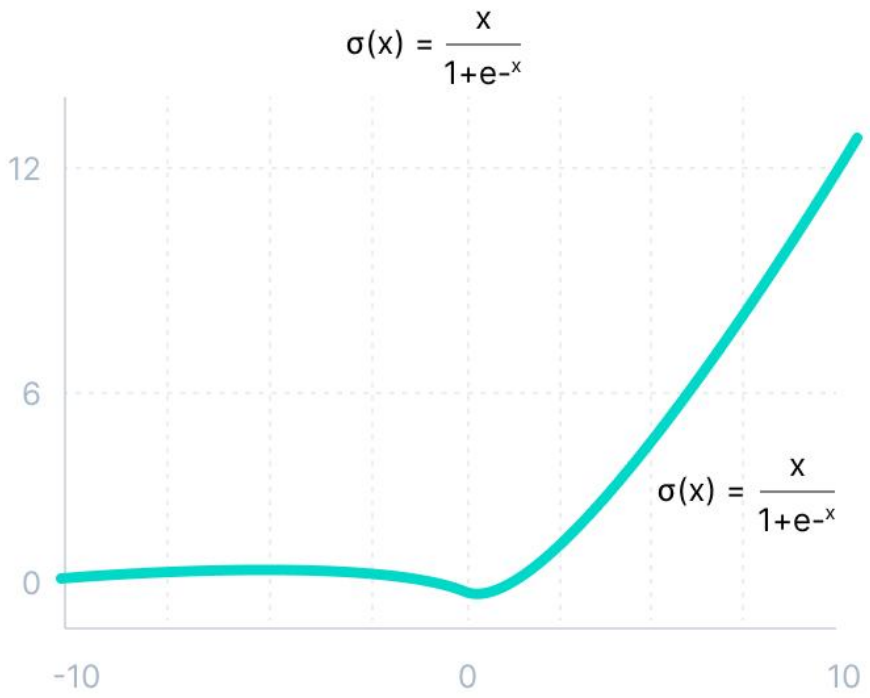
Функцијата враќа 1 за најголемиот индекс на веројатност додека враќа 0 за другите два индекси на низа. Овде, давајќи им целосна тежина на индексот 0 и без тежина на индексот 1 и индексот 2. Значи, излезот би бил класата што одговара на првиот неврон (индекс 0) од три.

На овој начин софтмакс функцијата помага решавање на класификациски проблеми кога имаме повеќе можни класи.

4.2.1.11. Свиш (Swish)

Тоа е само-ограничена функција за активирање развиена од истражувачи во Google.

Swish конзистентно одговара или ја надминува во перформанси функцијата за активирање ReLU на длабоки мрежи при примена на проблеми од различни домени како што се класификација на слики, машинско преведување итн. Графикот е даден на слика 46.



Слика: 46

Оваа функција е ограничена од долу, но неограничена од горе, т.е. *Y* се приближува до константна вредност кога *X* се приближува до негативната бесконечност, но *Y* се приближува до бесконечноста кога *X* се приближува до бесконечноста.

Математички може да се претстави како:

$$f(x) = x*sigmoid(x)$$

Еве неколку предности на функцијата за активирање Swish во однос на ReLU:

- Swish е мазна функција што значи дека не ја менува нагло насоката како што тоа го прави ReLU во близина на x = 0. Наместо тоа, непречено се наведнува од 0 кон вредностите < 0 и потоа повторно нагоре.
- Малите негативни вредности се нула во функцијата за активирање ReLU. Сепак, тие негативни вредности сè уште може да бидат релевантни за наоѓање шаблони во основата на податоците. Големите негативни вредности се анулираат.
- Функцијата swish што е немонотона го подобрува изразувањето на влезните податоци и тежината што треба да се научи.

4.2.1.12. Гаусова линеарна единица за грешка (GELU)

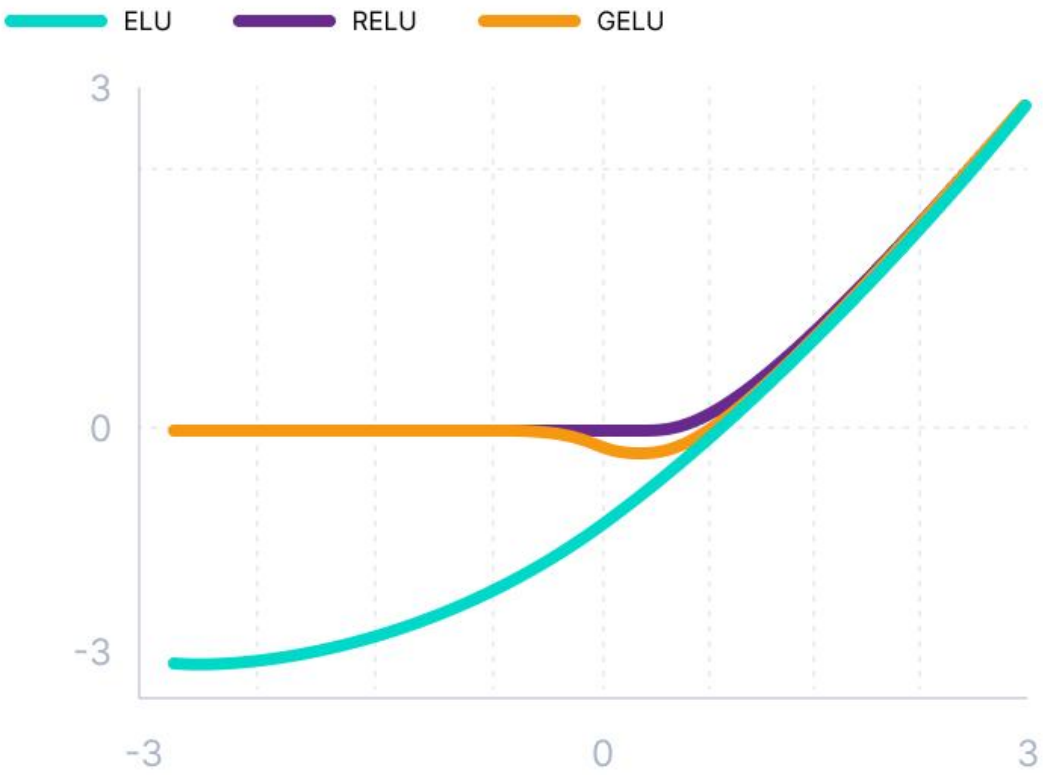
Активациската функција линеарна единица за Гаусова грешка (GELU) е компатибилна со BERT, ROBERTa, ALBERT и други врвни модели на NLP. Оваа функција за активирање е мотивирана со комбинирање на својствата на dropout, zoneout и ReLu. Dropout или напуштање е кога одреден неврон не пропагира излез во дадена итерација. Zoneout или доцнење е кога невронот ја пропагира повторно излезната вредност од претходната итерација.

ReLU и напуштањето заедно даваат излез на невронот. ReLU го прави тоа детерминистички со множење на влезот со нула или еден (во зависност од тоа дали влезната вредност е позитивна или негативна) и напуштање со стохастичко множење со нула.

Регулаторот RNN наречен zoneout стохастично ги множи влезовите со еден.

Ја спојуваме оваа функционалност со множење на влезот со нула или со еден што е стохастично определен и зависи од влезот. Влезот на невронот x го множиме со $m \sim Бернули(\Phi(x))$, каде што $\Phi(x) = P(X \leq x)$, $X \sim N(0, 1)$ е кумулативната дистрибуциска функција на стандардната нормална распределба.

Оваа дистрибуција е избрана бидејќи влезовите на невроните имаат тенденција да следат нормална дистрибуција, особено со нормализација на серијата. На слика 47 е дадена споредба на трите функции.



Слика: 47

Математички може да се претстави како:

$$f(x) = xP(X \leq x) = x\Phi(x)$$
$$= 0.5x\left(1 + tanh\left[\sqrt{2/\pi}\left(x + 0.044715x^3\right)\right]\right)$$

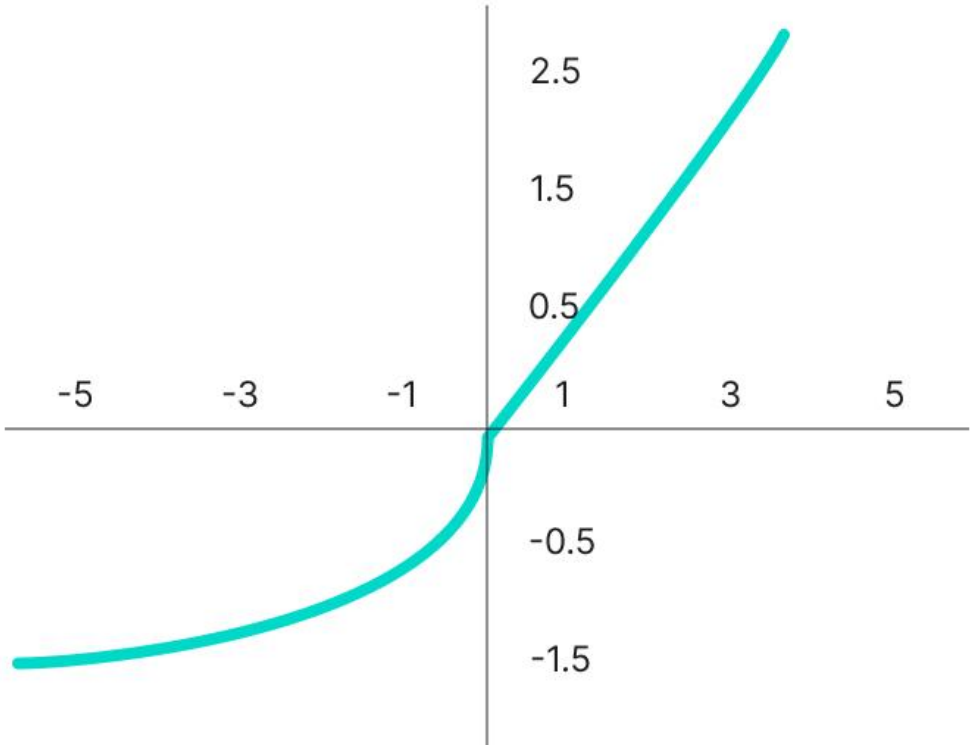
Нелинеарноста на GELU е подобра од активациите ReLU и ELU и покажува подобрувања во перформансите во сите задачи во домени на компјутерска визија, обработка на природни јазици и препознавање говор.

4.2.1.13. Скалирана експоненцијална линеарна единица (SELU)

SELU е дефинирана во мрежи за самонормализација и се грижи за внатрешна нормализација што значи дека секој слој ја зачувува средната вредност и варијансата од претходните слоеви. SELU ја овозможува оваа нормализација со прилагодување на средната вредност и варијансата.

SELU има и позитивни и негативни вредности за поместување на средната вредност, што беше невозможно за функцијата за активирање на ReLU бидејќи не може да даде негативни вредности.

Градиентите може да се користат за прилагодување на варијансата. На функцијата за активирање и треба регион со градиент поголем од еден за да се зголеми. На слика 48 е даден графикот на функцијата.



Слика: 48

Математички може да се претстави како:

$$f(\alpha, x) = \lambda \begin{cases} \alpha\left(e^{x} - 1\right) & for\ x < 0 \\ x & for\ x \geqslant 0 \end{cases}$$

SELU има предефинирани вредности на алфа α и ламбда λ.

Главна предност на SELU пред ReLU е тоа што внатрешната нормализација е побрза од надворешната нормализација, што значи дека мрежата конвергира побрзо.

SELU е релативно понова функција за активирање и има потреба од повеќе истражување за архитектури како што се CNN и RNN.

4.2.1.14. Предизвици

Постојат два предизвици кои често се среќаваат при тренирање на длабоки невронски мрежи.

4.2.1.14.1. Градиенти кои исчезнуваат

Како сигмоидна функција, одредени функции за активирање притискаат доволно влезен простор во мал излезен простор помеѓу 0 и 1. Затоа, голема промена во влезот на сигмоидна функција ќе предизвика мала промена во излезот. Оттука, дериватот станува мал. За плитки мрежи со само неколку слоеви кои ги користат овие активации, ова не е голем проблем. Меѓутоа, кога се користат повеќе слоеви, тоа може да предизвика градиентот да биде премногу мал за обуката да работи ефективно.

4.2.1.14.2. Експлодирачки градиенти

Експлодирачките градиенти се проблем каде што се акумулираат значителни градиенти на грешки и резултираат со многу големи ажурирања на тежините на моделите на невронската мрежа за време на обуката. Експлодирачки градиенти може да резултираат со нестабилни мрежи и учењето не може да се заврши. Вредностите на тежините исто така може да станат толку големи што да се прелеат и да резултираат со така наречени NAN (Not A Number) вредности.

Како да ја изберете вистинската функција за активирање?

Треба да ја усогласите вашата функција за активирање за вашиот излезен слој врз основа на видот на проблемот со предвидување што го решавате - конкретно, типот на предвидената променлива.

Најчесто, се започнува со користење на ReLU активациска функција и доколку таа не обезбеди задоволителни резултати, потоа се прават обиди со други функции. Дополнително, ReLU активациската функција треба да се користи само во скриените слоеви.

Функциите Sigmoid/Logistic и Tanh не треба да се користат во скриени слоеви бидејќи го прават моделот поподложен на проблеми за време на обуката (поради градиенти кои исчезнуваат).

Swish функцијата се користи во невронски мрежи со длабочина поголема од 40 слоеви.

Од аспект на проблем кој треба да се реши, постојат неколку препораки:

Регресија - Функција за линеарна активација

Бинарна класификација — Сигмоидна/логистичка функција за активирање

Класификација на повеќе класи - Softmax

Функцијата за активирање што се користи во скриените слоеви обично се избира врз основа на типот на архитектурата на невронската мрежа.

Конволуциски невронски мрежи (CNN): Функција за активирање ReLU.

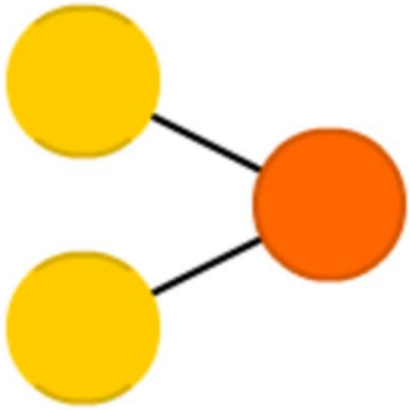
Рекурентна невронска мрежа: Tanh и/или Сигмоидна функција за активирање.

4.3. Типови невронски мрежи

Во зависност од тоа како невроните се поврзани меѓу себе, постојат голем број различни топологии на невронски мрежи. Од топологијата на мрежата во најголема мера зависи нејзината ефикасност и адекватност за решавање на специфичен проблем.

4.3.1. Перцептрон

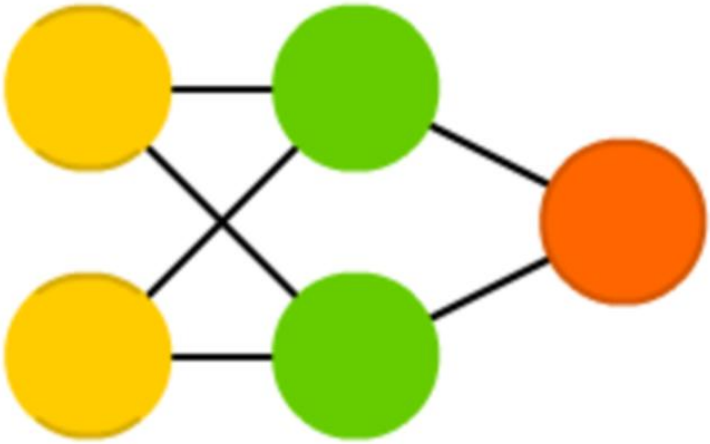
Најпростиот тип на невронска мрежа. Се состои од еден неврон: прима влезни параметри, ги сумира, ја аплицира активациската функција на сумата и ги враќа на излез. Шемата е дадена на слика 49.



Слика: 49

4.3.2. Невронска мрежа со предавање напред

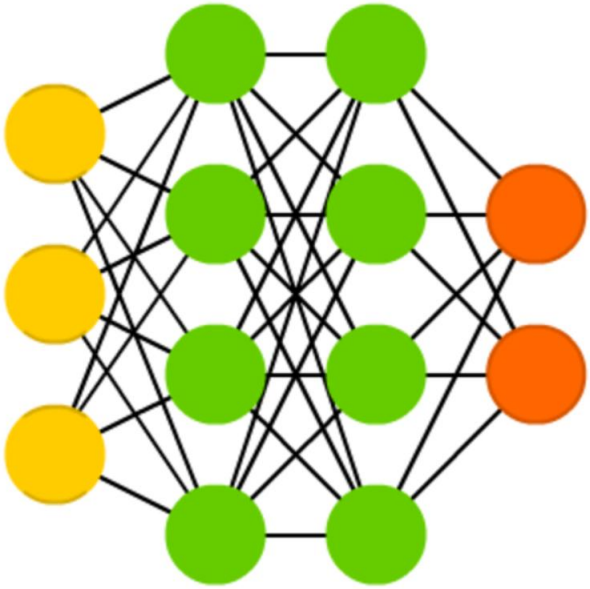
Моделот на оваа мрежа[33] е исто така многу стар, потекнува од 50тите години. Неколку клучни одлики на оваа мрежа се тоа што сите јазли се поврзани меѓу себе, активацијата се случува во една насока и постои само еден скриен невронски слој – односно еден слој неврони помеѓу влезниот и излезниот слој. Шемата е дадена на слика 50.



Слика: 50

4.3.3. Длабока невронска мрежа со предавање напред

Во суштина тоа е невронска мрежа со предавање напред но со повеќе од еден скриен слој. Постоењето на повеќе од еден скриен слој е клучната особина која и дава моќ на оваа мрежа и која ја врати релевантноста на невронските мрежи и вештачката интелигенција со воспоставувањето на длабокото учење како дисциплина. Шемата е дадена на слика 51.



Слика: 51

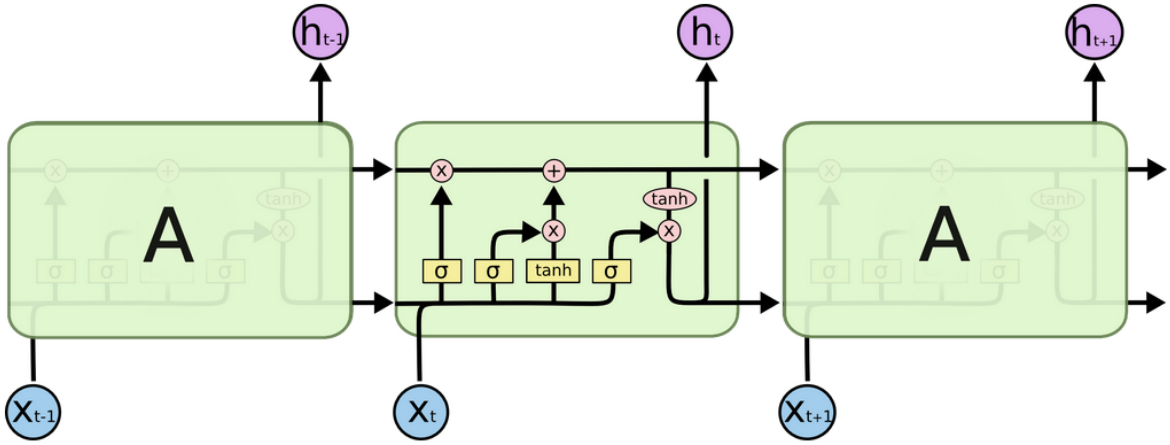
4.3.4. Рекурентни невронски мрежи

Тоа се во принцип длабоки мрежи со предавање напред, но дополнително, скриените неврони (невроните од скриените слоеви) се поинаков тип: рекурентни[35] келии. Рекурентните јазли освен класичните влезови од претходниот невронски слој, дополнително го земаат и сопствениот излез како влез со извесно задоцнување. Ова отвара простори за допонителна параметризација на мрежата и дава далеку подобри резултати во споредба со мрежите без рекурентни келии кога се решаваат проблеми каде е битен контекстот, односно одлуките од претходните изминувања имаат влијание на моменталното изминување. Оваа одлика им отвара примена во процесирањето на текст.

4.3.5. Невронски мрежи со продолжена кратка меморија

Овие мрежи се слични на рекурентните со тоа што рекурентните келии се заменети со нов тип на келии: мемориски келии[39]. Мемориските келии се на некој начин слични со рекурентните, но покомплексни и со тоа помоќни. Овие јазли може да одржуваат зависности подолго време, што решава еден од проблемите на рекурентните мрежи. Принципот на долготрајноста на меморијата е тоа што информациите што проаѓаат низ келијата подложат на математички операции кои ефективно може да ја маркираат информацијата како помалку или повеќе релевантна, односно келиите имаат способрност за селективно памтење и заборавање. На слика 52 е дадена шема од мемориски јазол. На дијаграмот се претставени портите во внатрешноста на невронот. Овие порти самите имаат влезови, излези и активациски функции кои управуваат со меморијата на

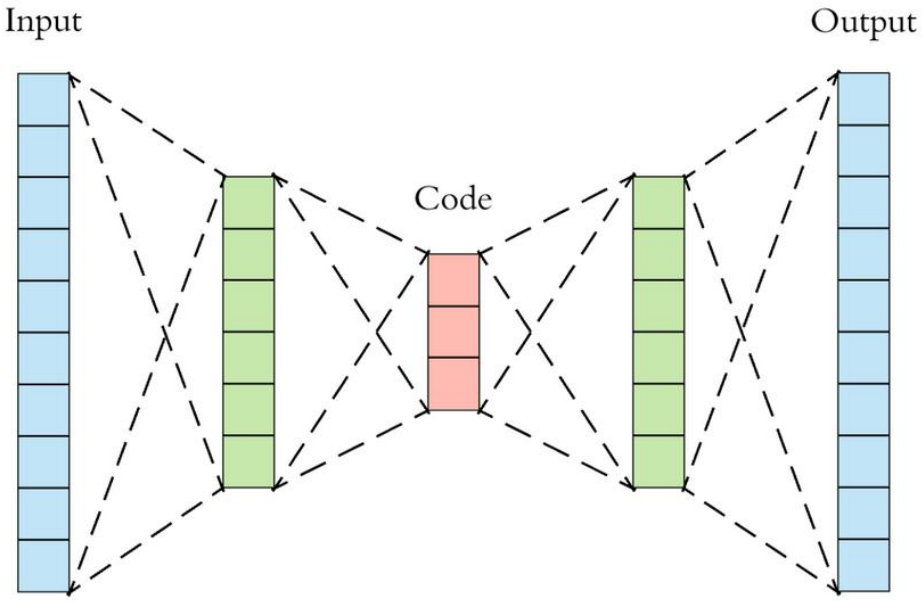
келијата.



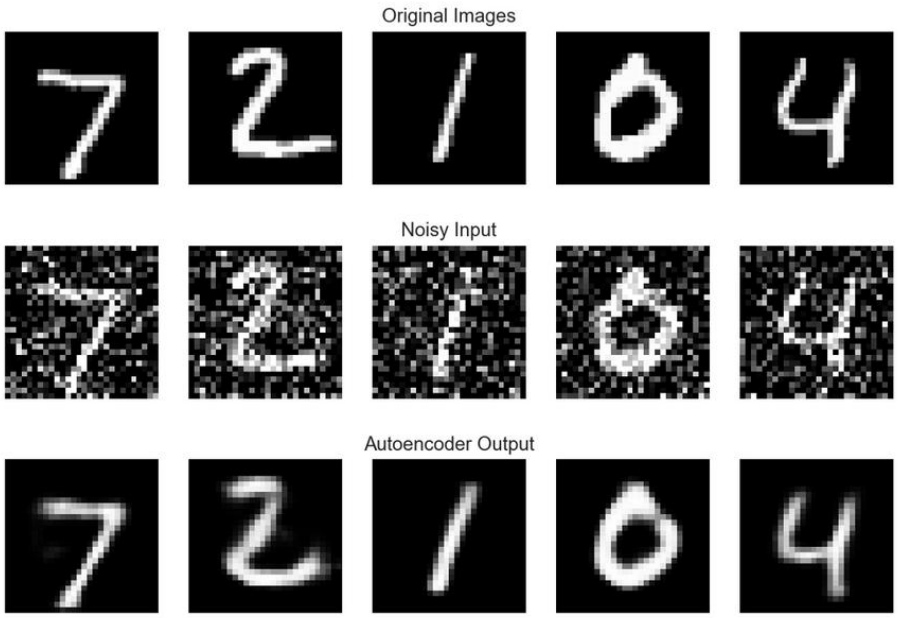
Слика: 52

4.3.6. Автоенкодери

Автоенкодерите[36] се комплетно поврзани невронски мрежи кои имаат влезен и излезен слој со ист број на неврони а скриените слоеви се различна димензија. Автоенкодерите спаѓаат во мрежи со учење без надзор. На слика 53 е дадена шема на автоенкодер за компресија. Најкрајните слоеви се со иста димензија а скриените имаат се помал број на неврони. Податоците што доаѓаат на влез се енкодираат до слојот со најмалку јазли, а потоа се декодираат до излезот. Ако автоенкодерот е добро истриниран, податокот на влез би бил многу сличен со податокот на излез. Можна примена на ваквите мрежи е компресија на податоци и отстранување на шум. На слика 54 е даден пример за работа на автоенкодер за отстранување на шум. Автоенкодерите може да имаат и структура таква што скриените слоеви имаат поголем број на јазли споредено со влезниот и излезниот слој, но не сите неврони работат во исто време и не секогаш истите неврони се неактивни.



Слика: 53



Слика: 54

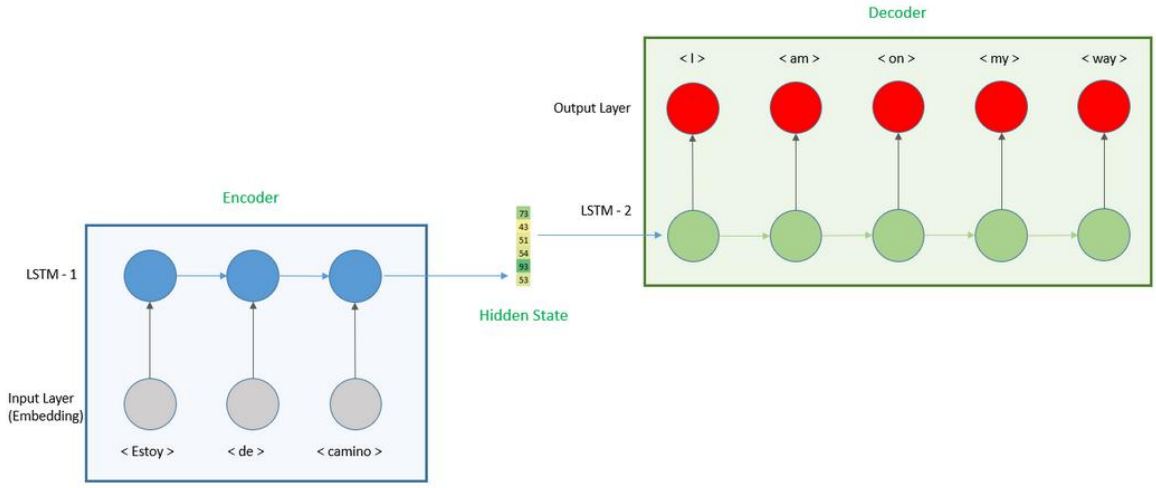
4.4. Композитни невронски мрежи

Со користење на еден тип на невронска мрежа може да се решат низа едноставни проблеми доста ефикасно, но за комплексни проблеми, како превод на текст, чет бот и слично, најчесто вакви едноставни модели не носат задоволителни резулати. За решавање на комплексни

проблеми се користат покомплексни архитектури кои кобинираат различни типови на невронски мрежи поврзани на различни начини. Ваквите системи имаат голем број параметри кои може да се подесат за зголемување на ефикасноста за решавање на конкретен проблем.

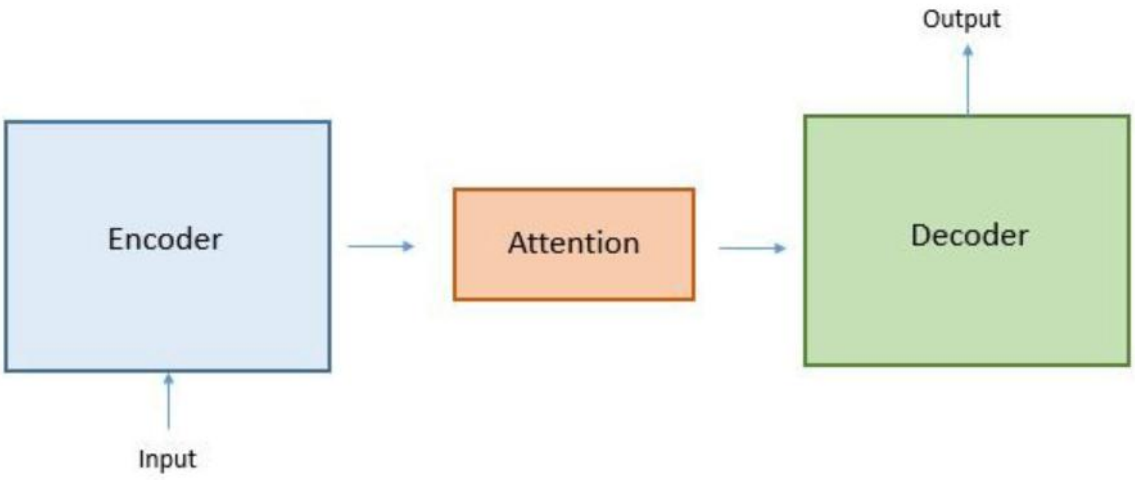
4.4.1. Енкодер/Декодер

Енкодер е модул кој во основа содржи две компоненти: влезен слој и LSTM (невронска мрежа со долга кратка меморија). Влезниот параметар на овој модул е текст а излезот е вектор. Декодерот е обратен од енкодерот: влезниот параметар е вектор и тој се пушта на LSTM мрежата прво, потоа на излезниот слој од кој што излезниот параметар е текст. Ако контруираме модел од енкодер и декодер, текстот кој се праќа на енкодерот се симплифицира во само еден вектор што значи дека моделот има проблеми при процесирањето на подолг текст – што е и познат проблем со LSTM мрежите. На слика 55 е дадена шемата.



Слика: 55

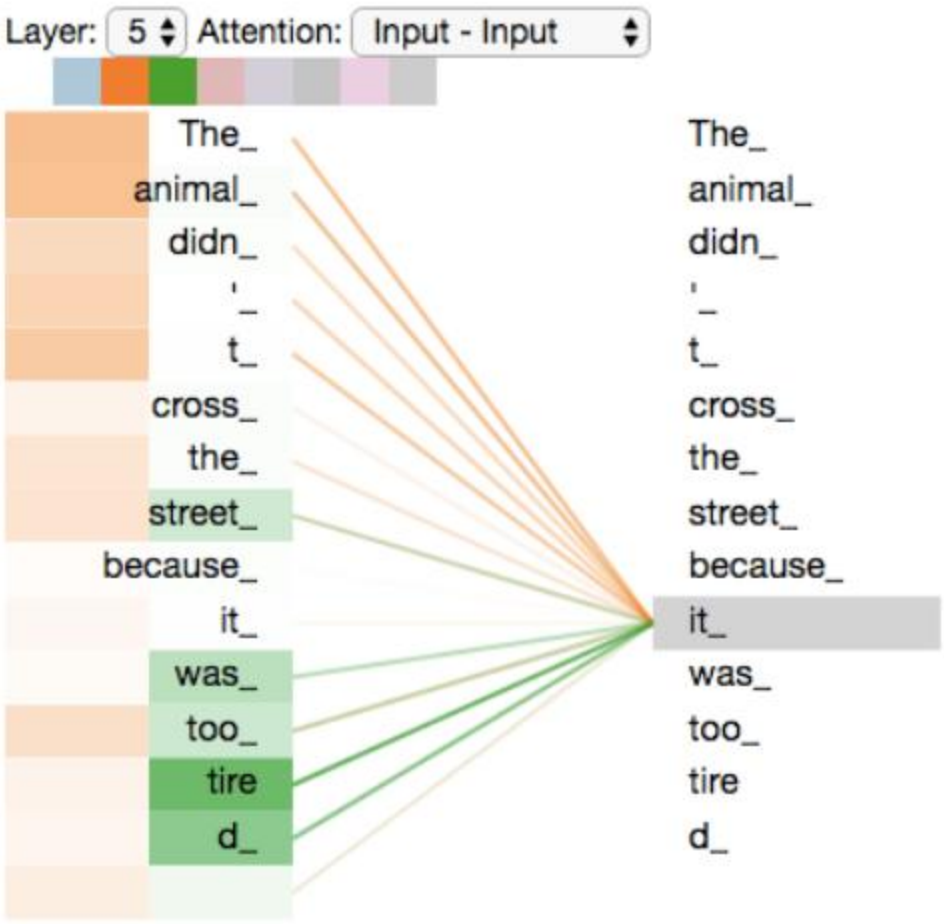
Еден начин да се реши тоа е да се користат повеќе енкодери. Тогаш излезот од таков модул би било матрица наместо вектор. Оваа матрица може да биде од големи димензии и тогаш се јавува обратниот проблем: премногу информации стигаат до декодерот.



Слика: 56

Како решение за проблемот се конструира поимот за внимание. Вниманието е математичка операција која се аплицира врз матрицата така што дел од вредностите добиваат поголема а дел помала тежина. Конкретната финкција зависи од проблемот кој се решава. Бидејќи вниманието се аплицира помеѓу енкодерите и декодерите, во еден модел може да има имплементација на само една инстанца од вниманието. На ваков модел му недостасува ширина. На слика 56 е дадена шемата.

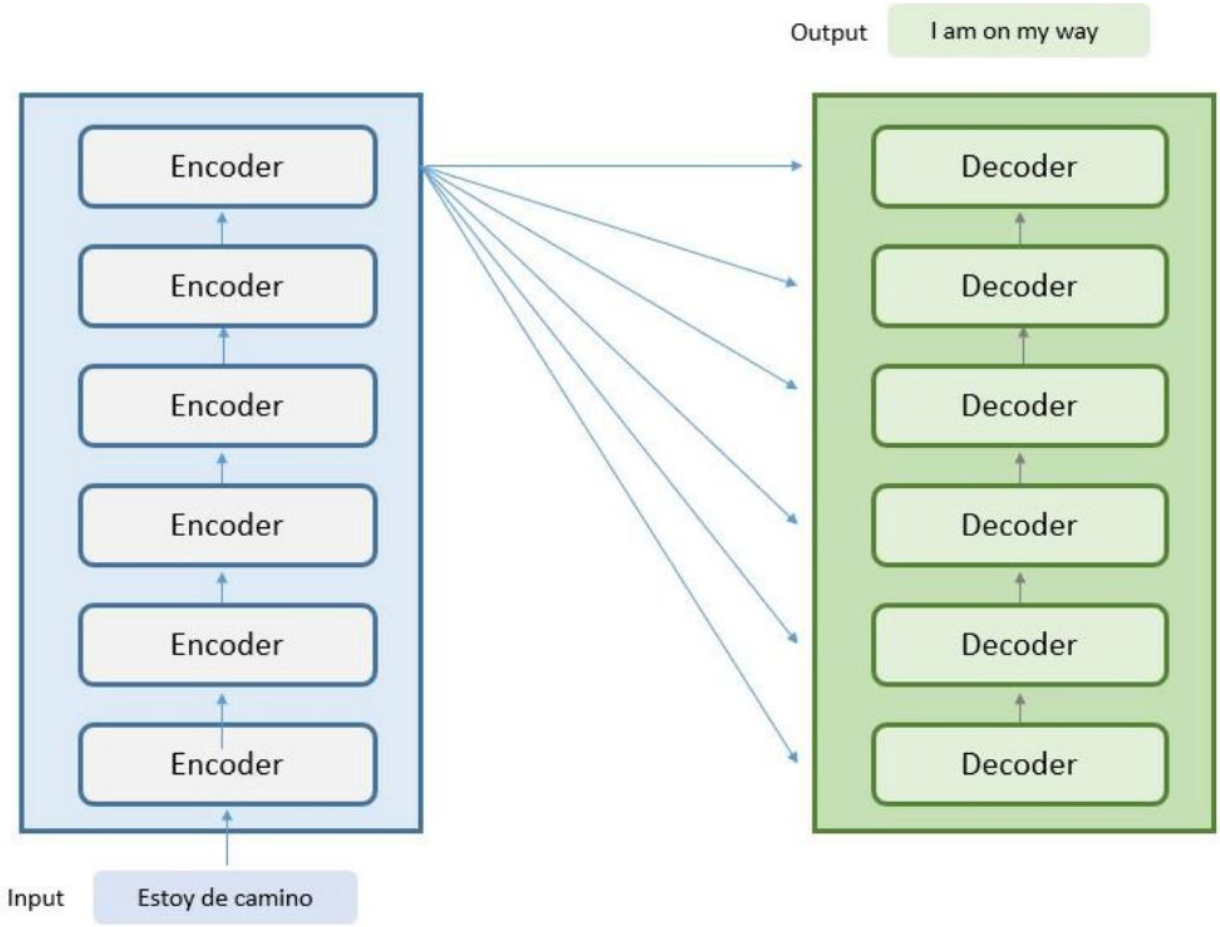
Решението е внимание со повеќе „глави“. На слика 57 е дадена илустрација како би изгледало такво внимание. Во принцип, ако се аплицираат повеќе глави на иста секвенца од текст се додава ширина на моделот, односно еден токен од текстот може да се асоцира со повеќе други. Примерот илустрира реченица „ Животното не ја помина улицата бидејќи тоа беше преморено“, со едно внимание може да се поврзе „тоа“ со „животното“. Но ако се додаде и втора глава, тогаш се овозможува асоцијација и со други токени од реченицата, пример „преморено“. Имплементација на внимание со повеќе глави се имплементира преку трансформери.



Слика: 57

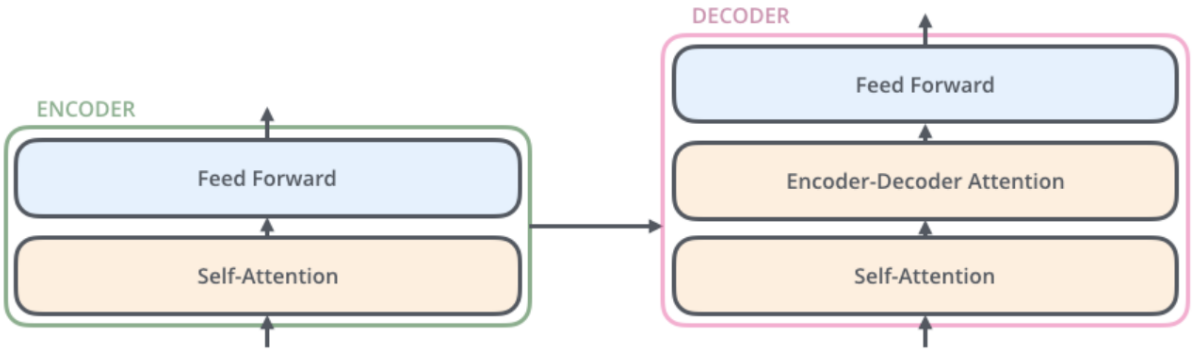
4.4.2. Трансформери

Трансформерите[37] се модули кои се базираат на две составни компоненти: енкодер и декодер. Спецификите околу конкретната имлементација на енкодерите и декодерите внатре во трансформерот може да биде различна. Најчесто во еден трасформер[20] има повеќе декодери и енкодери, секој од нив често се нарекува трансформерски слој. Бројот на трансформерски слоеви исто така е различен кај различните модели.



Слика: 58

На слика 58 е дадена структурата на енкодер и декодер во склоп на трансформер. Во најопшта смисла, секој трансформерски слој има енкодер во кој има имплементација на внимание и невронска мрежа со предавање напред; декодер со имплеменација на внимание кон влезот од енкодерот и внимание на конекст и невронска мрежа со предавање напред. Зависно од конкретната имплементација можно е да постојат и дополнителни компоненти: за нормализација, аплицирање на софт макс и слично. На слика 59 е дадена шемата.

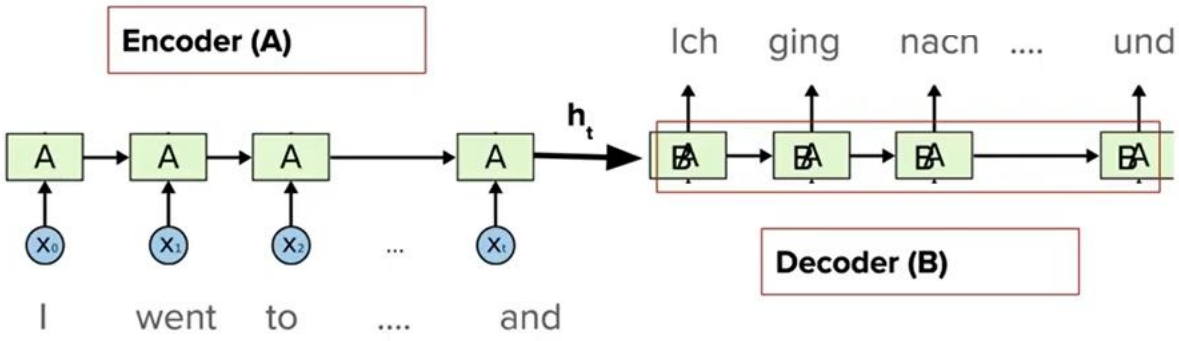


Слика: 59

Трансформерите се доста често користени како основа за изградба на модели со висока ефикасност. Постојат неколку доста познати трансформери како GPT[17], BART[24] и BERT[14].

4.4.3. BART

Автоенкодер за отстранување на шум кој е трениран со корупција на документи и нивна реконструкција. Моделот има две верзии, со 6 и со 12 слоја на енкодери и декодери. BART користи класичен секвенца кон секвенца модел (seq2seq). Секвенца кон секвенца моделот е комбинација од две невронски мрежи каде едната на зема цела секвенца текст го енкодира низ скриените слоеви и еден од скриените излези (излез од скриен слој) се пушта како влез на декодерот кој пак е посебна невронска мрежа што го декодира тој влез и го пушта на излез како секвенца од текст. На слика 60 е дадена шема за seq2seq модел.

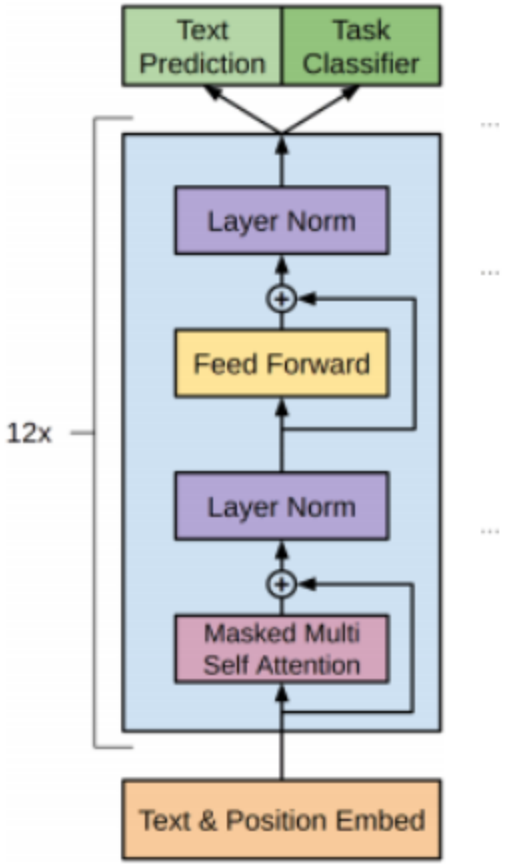


Слика: 60

BART е доста ефикасен модел и може да се користи за решавање на повеќе различни проблеми: реконструкција на оштетен документ односно генерирање на текст, сумаризација на текст, дијалог и превод.

4.4.4. GPT

Generative Pre-Training, GPT[15] е NLP модел базиран на трансформер. GPT има 12 слоеви трансформатори, секој со 12 независни механизми за внимание; резултатот е 12 x 12 = 144 различни шеми на внимание. Секој од овие шеми на внимание одговара на лингвистичката особина доловена од моделот. На слика 61 е дадена архитектурата на GPT.



Слика: 61

GPT е трениран како класичен јазичен модел да предвидува следен збор во даден контекст. Нема многу добри перформанси. Неговиот следбеник GPT2 е буквално само скалиран GPTсо 10 пати повеќе параметри и 10 пати повеќе податоци, но има далеку поголема ефикасност. GPT-2 е голем јазичен модел базиран на трансформер со 1,5 милијарди параметри, трениран на база на податоци од 8 милиони веб-страници. Клучно за ефикасноста на овој модел е начинот на кој е трениран. Имено, податоците се исклучително содржини напишани и едитирани од човек со висок квалитет и голема разновидност. Тимот кој го разви овој модел изјави дека нема да биде објавен целиот трениран модел заради можноста за злоупотреба. Текстовите што ги генерира GPT2 изгледаат сосема природно и може да се користат за генерирање на лажни вести од најразличен домен. Бидејќи квалитетот на множеството за тренинг е на високо ниво, текстовите што се генерираат не само што имаат смисла, туку имаат проектираат и стил на пишување. Генериран текст за позната личност навистина звучи како да е напишано од таблоид, текст за вештачка интелигенција го има вокабуларот на научен труд. Подолу се дадени неколку примери за илустрација на моќта на моделот, преземени од официјалната страница.

>>Model Completion (machine-written, first try)

Here you can see the most recent progress with Generative Pre-trained Transformer:

Figure 1: Generative Pre-trained Transformer training on several texts.

We are now preparing a collection of datasets for translation and machine translation in our language model. We will be using one of the large number of text samples provided by The New York Times.

…<<

>>System Prompt (human-written)

Miley Cyrus was caught shoplifting from Abercrombie and Fitch on Hollywood Boulevard today.

Model Completion (machine-written, second try)

The 19-year-old singer was caught on camera being escorted out of the store by security guards.

The singer was wearing a black hoodie with the label 'Blurred Lines' on the front and 'Fashion Police' on the back.

Scroll down for video

Shoplifting: Miley Cyrus was caught shoplifting from Abercrombie and Fitch on Hollywood Boulevard today (pictured)

The singer was wearing a black hoodie with the label 'Blurred Lines' on the front and 'Fashion Police' on the back

The singer was also wearing a pair of black-rimmed glasses, a black jacket, black jeans and black sandals.

She was carrying a pair of black and white striped gloves and a small black bag. <<

>>System Prompt (human-written)

Legolas and Gimli advanced on the orcs, raising their weapons with a harrowing war cry.

Model Completion (machine-written, first try)

The orcs' response was a deafening onslaught of claws, claws, and claws; even Elrond was forced to retreat. "You are in good hands, dwarf," said Gimli, who had been among the first to charge at the orcs; it took only two words before their opponents were reduced to a blood-soaked quagmire, and the dwarf took his first kill of the night. The battle lasted for hours until two of the largest Orcs attempted to overwhelm Aragorn. When they finally stopped, they lay defeated and lifeless for miles and miles. <<

4.4.5. BERT

Започнувајќи со моделската архитектура на трансформаторот, ги дефиниравме градежните блокови на архитектурата на оригиналниот Трансформер. Да го замислиме оригиналниот трансформатор како модел изграден со коцки. Градежниот сет содржи тули како што се енкодери,

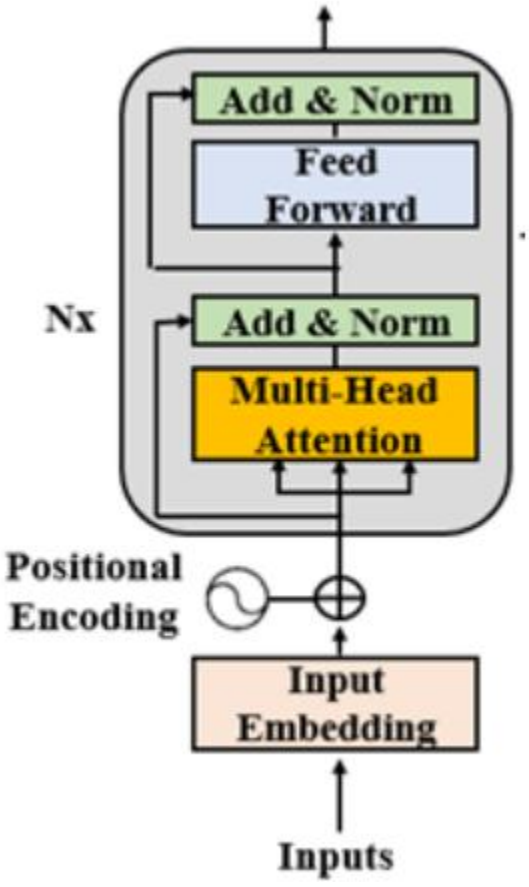
декодери, слоеви за вградување, позиционо кодирање методи, слоеви на внимание со повеќе глави, слоеви за внимание со маскирани повеќе глави, нормализациски слој, под-слоеви за повлекување и линеарни излезни слоеви. Со овие градежни блокови може да се изработат различни конструкции.

BERT додава ново парче во комплетот за градење на Transformer: двонасочен подслој со внимание со повеќе глави. Кога ние луѓето имаме проблеми со разбирањето на една реченица, не гледаме само на минатите зборови. БЕРТ, како нас, гледа на сите зборови во иста реченица во исто време. BERT ги користи само блоковите на енкодери на трансформерот на нов начин и не го користи стекот на декодери

4.4.5.1. Архитектурата на BERT

BERT воведува двонасочно внимание на моделите на трансформери. Двонасочното вниманието бара многу други промени на оригиналниот модел Transformer.

Првиот градежен блок земан од оригиналниот модел на Transformer е енкодерскиот слој. Слојот на енкодери е прикажан на слика 62.



Слика: 62

Моделот BERT не користи слоеви на декодер. Моделот BERT има стек на енкодери но нема стек на декодери. Маскираните токени (ги кријат токените за да се предвидат) се во слоевите на внимание на енкодерот.

Оригиналниот трансформатор содржи куп од N=6 слоеви. Бројот на димензии на оригиналниот трансформер е d = 512. Бројот на глави за внимание на оригиналниот трансформер е A=8. Димензиите на главата на оригиналниот трансформер се:

$$d_k = \frac{d_{model}}{A} = \frac{512}{8} = 64$$

Слоевите на BERT енкодерот се поголеми од оригиналниот модел на трансформер. Два BERT модели може да се изградат со слоевите на енкодер:

• BERT-base, кој содржи куп од N=12 енкодерски слоеви. d = 768 и може исто така да се изрази како H=768, како во трудот BERT. Подслојот за внимание со повеќе глави содржи A=12 глави. Димензиите на секоја глава z остануваат 64 како во оригиналниот модел на трансформер:

$$d_k = \frac{d_{model}}{A} = \frac{768}{12} = 64$$

Излезот на секој подслој за внимание со повеќе глави пред спојувањето ќе биде излезот од 12-те глави:

output_multi-head_attention={z0, z1, z2,...,z11}

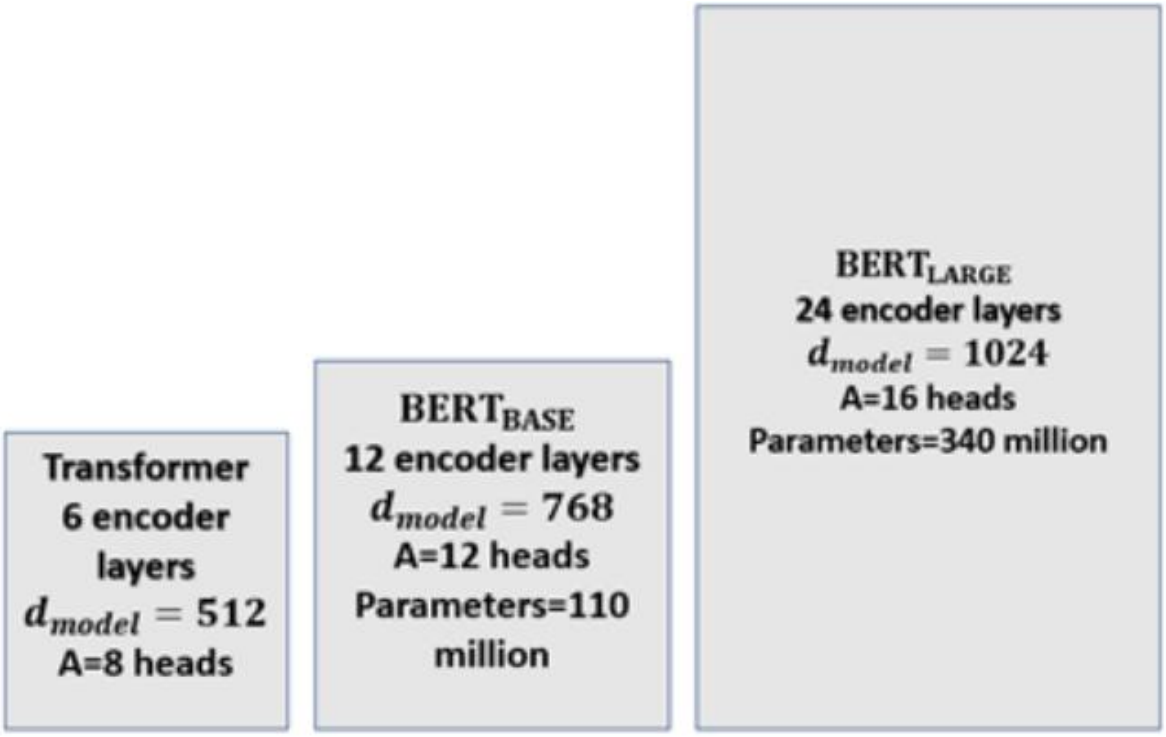
• БЕРТ-large, кој содржи куп од N=24 слоеви на енкодер. d = 1024. Подслојот за внимание со повеќе глави содржи A=16 глави. Димензиите на секоја од главите z исто така останува 64 како во оригиналниот модел на Transformer:

$$d_k = \frac{d_{model}}{A} = \frac{1024}{16} = 64$$

Излезот на секој подслој за внимание со повеќе глави пред спојувањето ќе биде излезот од 16-те глави:

output_multi-head_attention={z0, z1, z2,...,z15}

Визуелно, големините на моделите се прикажани на слика 63.



Слика: 63

Големината и димензиите играат суштинска улога во предобука во стилот на БЕРТ. Моделите на BERT се како луѓе. Моделите BERT даваат подобри резултати со поголема работна меморија (димензии), и повеќе знаење (податоци). Модели на големи трансформатори кои учат од големи количини на податоци ќе бидат подобро пред-тренирани за извршување на подоцнежните NLP задачи.

Моделот BERT нема куп слоеви од декодер. Како таков, нема маскиран подслој за внимание со повеќе глави. Маскиран слој на внимание со повеќе глави ги маскира сите токени кои се надвор од сегашната позиција. На пример,во реченицата:

Мачката седна на него бидејќи беше убав килим.

Ако моменталната итерација е на зборот „него“, влезот на енкодерот може да биде:

Мачката седна на него <маскирана секвенца>

Мотивацијата на овој пристап е да се спречи моделот да го види резултатот што би требало да го предвиди. Овој пристап од лево кон десно дава релативно добри резултати.

Сепак, моделот не може многу да научи на овој начин. За да знаеме на што се однесува „него“, треба да се види целата реченица за да се обработи зборот „килим“ за да стане јасно дека „него“ се однесува на килимот.
Ова е токму еден од начините на кој е трениран BERT моделот. Моделот е обучен со две задачи. Првиот метод е Моделирање со Маскиран Јазик (MLM). Вториот метод е предвидување на следната реченица (NSP).

4.4.5.1.1. Моделирање со маскиран јазик

Моделирањето со маскиран јазик не бара обука на модел со низа од видливи зборови проследени со маскирана низа за предвидување. BERT воведува двонасочна анализа на реченица со маска на случаен збор од реченицата.

Потенцијална влезна низа може да биде:

„Мачката седна на него бидејќи беше убав килим.“

Декодерот ќе ја маскира низата на внимание откако моделот ќе го достигне зборот

"него":

„Мачката седна на него <маскирана секвенца>.“

Но, BERT енкодерот маскира случаен токен за да направи предвидување:

„Мачката седна на него [МАСКА] беше убав килим“.

Подслојот со повеќе внимание сега може да ја види целата низа, да го изврши само-вниманието и да го предвиди маскираниот токен. Влезните токени се маскираат на неправилен начин за да го принудат моделот да тренира подолго, но да постигне подобри резултати со три методи:

• Изненадувајќи го моделот со тоа што нема да се маскира ниту еден токен на 10% од сетот на податоци, пример:

„Мачката седна на него [бидејќи] беше убав килим.“

• Изненадувајќи го моделот со замена на токенот со случаен токен на 10% од базата на податоци, на пример:

„Мачката седна на него [често] беше убав килим.“

• Заменувајќо токен со маскиран токен на 80% од сетот на податоци, на пример:

„Мачката седна на него [МАСКА] беше убав килим.“

Овој пристап го избегнува проблемот со overfitting - што означува модел кој предобро функционира врз ограниченото множество податоци за тренинг, тој модел кога ќе се пушти во реалниот свет има далеку полоши перформанси. Користејќи го овој пристап, моделот се присилува да работи подолго, односно процесот на тренирање трае подолго, но се добива модел со голема ефикасност.

4.4.5.1.2. Предвидување на следната реченица

Вториот метод пронајден за обука на БЕРТ е предвидување на следната реченица (NSP). Влезот има две реченици. Додадени се два нови токени:

• [CLS] е токен за бинарна класификација додаден на почетокот на првата низа за да се предвиди дали втората низа ја следи првата низа. Позитивен примерок обично е пар последователни реченици земени од базата на податоци. Негативен примерок се создава со користење на секвенци од различни документи.

• [SEP] е токен за одвојување што го сигнализира крајот на низата.

На пример, влезните реченици извадени од книга може да бидат:

„Мачката спиеше на тепихот. Таа сака да спие цел ден.“

Овие две реченици ќе станат една влезна целосна низа:

[CLS] мачката спиеше на килимот [SEP] Таа сака да спие цел ден[SEP]

Овој пристап бара дополнителни информации за кодирање за да се разликува низата А од низата Б. На табелата е претставен процесот на енкодирање. Првата редица е влезот, втората е вградувањето на токените, третата вградувањето на реченициите и последната е позициското кодирање.

[CLS]	Мачката	спиеше	на	килимот	[SEP]	Таа	сака	да	спие	цел	Ден	[SEP]
E[cls]	E[мачката]	E[спиеше]	E[на]	E[килимот]	E[sep]	E[таал]	E[сака]	E[да]	E[спие]	E[цел]	E[ден]	E[sep]
+	+	+	+	+	+	+	+	+	+	+	+	+
E[A]	E[A]	E[A]	E[A]	E[A]	E[A]	E[B]	E[B]	E[B]	E[B]	E[B]	E[B]	E[B]
+	+	+	+	+	+	+	+	+	+	+	+	+
E[0]	E[1]	E[2]	E[3]	E[4]	E[5]	E[6]	E[7]	E[8]	E[9]	E[10]	E[11]	E[12]

Влезните вградувања се добиваат со сумирање на вградените токени, вградените сегменти (реченица, фраза, збор) и позициското кодирање. Подслојот за вградување на влез и позициско кодирање на моделот BERT може да биде сумиран на следниов начин:

• Низата зборови се дели на WordPiece токени.

• Токен [MASK] случајно ќе ги замени почетните токени за зборови за трениерање со маскирано јазично моделирање.

• [CLS] токен за класификација е вметнат на почетокот на низата со цел класифицирање.

• [SEP] токен одделува две реченици (сегменти, фрази) за обука за NSP.

• Вградувањето на реченицата се додава во вградувањето на токен, така што вградената реченицата А има различна вредност од вградената реченица Б.

• Позиционото кодирање се учи. Не се применува Синус-косинус методот на енкодирање на оригиналниот трансформер.

Некои дополнителни клучни карактеристики се:

• BERT користи двонасочно внимание во сите негови подслоеви за внимание со повеќе глави, што отвара нови хоризонти на учење и разбирање на односите помеѓу токените.

• BERT воведува сценарија на вградување без надзор, со тоа пред-тренира модели со нелабелиран текст. Ова го принудува моделот да размислува посилно за време на процесот на учење со внимание со повеќе глави. Ова го прави БЕРТ способен да научи како јазиците се градат и го применуваат ова знаење во идните задачи без да мора да се предтренира секој пат.

• BERT исто така користи учење со надзор, покривајќи ги сите основи во процесот за предтренирање.

BERT претставува нови принципи за тренирање на модели базирани врз трансформери. Постигнува одлични резултати на неколку различни унифицирани тестови. Во 2019 година BERT се става во употреба во Google Search се до 2021 кога е заменет со MUM моделот. BERT е трениран на википедиа и повеќе од 10 000 книги од различни жанрови. Временски за тренирање на

моделот потребни се до 3 дена користејќи TPU – чипови специјализирани за тренирање на ВИ модели. BERT е многу моќен модел и врз основа на базичната архитектура се конструирани повеќе различни верзии, направени да одговараат за специфични примени и задачи. Следува осврт на фамилијата BERT.

4.4.6. ALBERT

Како што беше наведено претходно, базата БЕРТ се состои од 110 милиони параметри што ја прави пресметковно интензивна и затоа беше потребна лесна верзија со намалени параметри. Моделот ALBERT[25] има 12 милиони параметри со 768 скриени слоеви и 128 слоеви за вградување. Како што е очекувано, полесниот модел го намалува времето за обука и времето за заклучување. За да се постигне помал сет на параметри, се користат следните 2 техники:

- Споделување на параметри меѓу слоеви: во овој метод, параметарот само на првиот енкодер се учи и истиот се користи кај сите енкодери. Овие параметри може да се споделат за:
    - Само слојот со невронската мрежа со предавање на напред
    - Слојот за предавање напред и слојот за внимани со повеќе глави
    - Само за слојот за внимани со повеќе глави
- Факторизирана параметаризација на слојот за вградување: Наместо да се задржи слојот за вградување на 768, слојот за вградување се намалува со факторизација на 128 слоеви.

Покрај тоа што АЛБЕРТ е лесен, за разлика од БЕРТ кој работи на NSP, АЛБЕРТ работи на концепт наречен SOP (Предвидување редослед на реченици). Клучната разлика помеѓу NSP и SOP е тоа што за NSP за време на обуката, моделот добива влезни парови реченици и учи да предвидува дали втората реченица е следната реченица и во оригиналниот текст. SOP е „модел на класификација“ каде што целта е да се „класифицира“ дали 2-те дадени реченици се заменети или не, т.е. дали се во правилен редослед.

4.4.7. RoBERTa

RoBERTa[16] е кратенка за „Робусно оптимизиран пристап за пред-тренирање на BERT“. На многу начини ова е подобра верзија на моделот BERT. Клучните точки на разлика се следните:

- Динамичко маскирање: BERT користи статичко маскирање, односно истиот дел од реченицата е маскиран во секоја епоха. Спротивно на тоа, RoBERTa користи динамично маскирање, при што за различни епохи различни дел од реченициte се маскирани. Ова го прави моделот поробустен.
- Отстранување на задачата NSP: Беше забележано дека задачата NSP не е многу корисна за пред-тренирање на моделот BERT. Затоа, RoBERTa се тренира само со задачата MLM.
- Поголемо множество податоци: BERT е пред-трениран на „Toronto BookCorpus“ и „Anglish Wikipedia databases“, односно вкупно 16 GB податоци. Спротивно на тоа, покрај овие две збирки на податоци, RoBERTa е обучен и на други збирки на податоци како CC-News (Common Crawl-News), Open WebText итн. Вкупната големина на овие збирки на податоци е околу 160 GB.
- Поголеми серии: за да ги подобри брзината и перформансите на моделот, во RoBERTa се користи серија од 8.000 итерации со 300.000 чекори. За споредба, кај BERT се користи серија од 256 итерации со 1 милион чекори.

4.4.8. ELECTRA

ELECTRA[26] е кратенка за „Ефикасно учење на енкодер кој прецизно ги класифицира заменетите токени“. Моделот користи генератор-дискриминаторска структура. Освен што е полесна верзија на BERT, ELECTRA ги има следните карактеристични карактеристики:

- Откривање на заменети токени: Наместо MLM за пред-тренинг, ELECTRA користи задача наречена „Откривање заменети токени“ (RTD). Во RTD, наместо маскирање на токенот, токенот се заменува со погрешен токен и се очекува моделот да се класифицира, без разлика дали токените се заменети со погрешни или не.
- Исто како кај RoBERTa, не се врши предобука за NSP.
- Подобро тренирање: Наместо да се тренира само за податоците за маската, кај моделот ELECTRA обуката се изведува на целосни податоци.

4.4.9. SpanBERT

SpanBERT[27] е специјална варијанта на BERT каде целта е да се користи моделот за одговарање на прашања, екстракција на релација итн. Ова не значи дека моделот може да го створи одговорот на прашањето, туку може да го најде, во смисла, да идентификува групата (распон) зборови во параграф (текст блок) кои се можен одговор на поставеното прашање. Интуитивно ова би значело дека на моделот БЕРТ ќе му требаат следните измени:

- Маскирање „распон на зборови“ наместо „случајно маскирање“ на токен. Ако земеме реченица за пример:
  Токени = [Податоци, наука, комбинации, домен, експертиза, програмирање, вештини и знаење за математика и статистика]
  Во случајот MLM, се врши случајно маскирање:
  Токени = [Податоци, наука, [МАСКА], домен, [МАСКА], програмирање, вештини, [МАСКА], знаење, за, математика, [МАСКА], статистика]
  Меѓутоа, за распон, маскирањето се прави по случаен избор, но на распон од последователни зборови:
  Токени = [Податоци, наука, комбинации, [МАСКА], [МАСКА], [МАСКА], [МАСКА], [МАСКА], знаење за, математика и статистика]
- Ажурирање на функцијата за загуба, така што на токените во „распонот на зборовите“ ќе им се даде соодветно значење. Покрај промената во логиката на маскирање, треба да се ажурира и функцијата на загуба за да ги земе предвид и двете:
  - MLM Цел: Предвиди го маскираниот токен, односно веројатноста сите зборови во вокабуларот да бидат маскираниот збор.
  - Цел на граници на распон (SBO): За SBO, наместо да се земе претставата на маскираниот токен, се земаат предвид оние на соседните токени. Во нашиот пример:
    Токени = [Податоци, наука, комбинации, [МАСКА], [МАСКА], [МАСКА], [МАСКА], [МАСКА], знаење за, математика и статистика]
    Претставувањето на зборовите „комбинации“ и „знаење“ ќе се користи за да се изведат маскираните токени. Ова дополнително имплицира дека кодираната позиција ќе се користи за разбирање на релативните позиции на токените.

Целокупната функција на загуба на SpanBERT = Загуба на целта на MLM + Загуба на целта на SBO

Моделот е обучен да ја минимизира оваа функција на загуба. Овој претходно обучен модел може да се користи за која било задача за одговарање на прашања или екстракција на релација.

4.4.10. DistilBERT и TinyBERT

За разлика од другите варијанти кои изменија некои аспекти на моделот BERT за да создадат нова верзија, и DistilBERT[28] и TinyBERT[29] работат на различен начин.

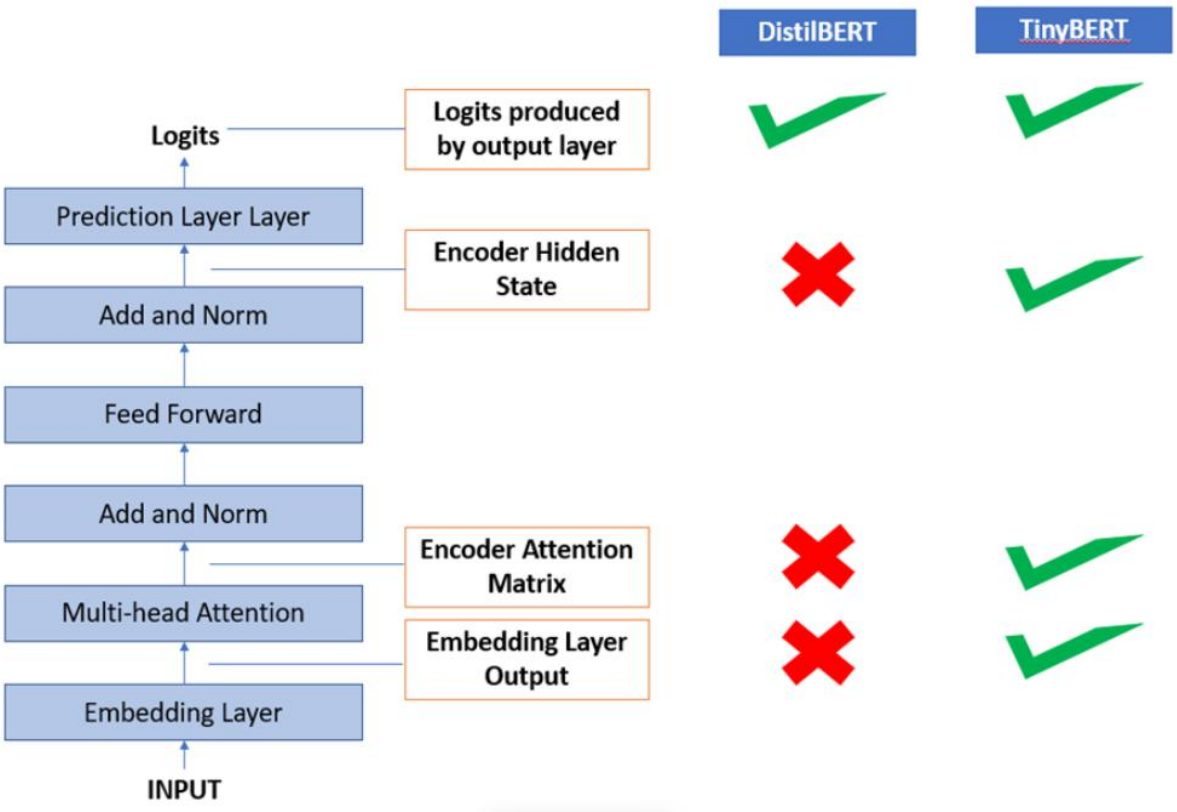
Овие варијанти не го менуваат моделот, туку се обидуваат да ја намалат големината на моделот со создавање на помал модел кој некако го реплицира излезот од поголемиот модел.

Процесот се нарекува дестилација на знаење, односно овие помали модели го дестилираат знаењето за големите модели и се обидуваат да го добијат сличниот исход на моделот со помала големина. Општата терминологија што се користи за поголемиот и помалиот модел е „ученик“ и „наставник“. Севкупно, во DistilBERT се користи претходно обучена мрежа (наставник) за да се обучи мрежата на учениците за да се добие знаење од мрежата на наставници преку дестилација.

TinyBERT е дополнително проширување каде што покрај дестилирањето на знаењето од излезниот слој, TinyBERT дестилира и информации од средните слоеви.

TinyBERT = DistilBERT + дестилирано знаење од посредничките слоеви

На слика 64 е дадена визуелна претстава како овие два модела работат.



Слика: 64

Функцијата за загуба на TinyBERT не само што ќе вклучува загуби од излезниот слој, туку и од слојот за внимание и слојот за вградување.

5. Машинско учење во облаците

За повеќето бизниси, машинското учење изгледа блиску до ракетната наука, изгледа скапо и бара талент. И, ако имате за цел да изградите друг систем за препораки на Netflix, тоа навистина е. Но, трендот на правење сè како услуга влијае и на оваа софистицирана сфера. Со ова, машинското учење и вештачката интелигенција се доближуваат поблиску до средните и мали бизниси и просечните поединци.

Една од најинспиративните приказни на МЛ е онаа за јапонски фармер кој решил автоматски да ги сортира краставиците за да им помогне на своите родители во оваа макотрпна операција. За разлика од приказните што изобилуваат за големите претпријатија, момчето не поседува експертиза за машинско учење, ниту голем буџет. Но, тој успеа да се запознае со TensorFlow и примени длабоко учење за да препознае различни класи на краставици.

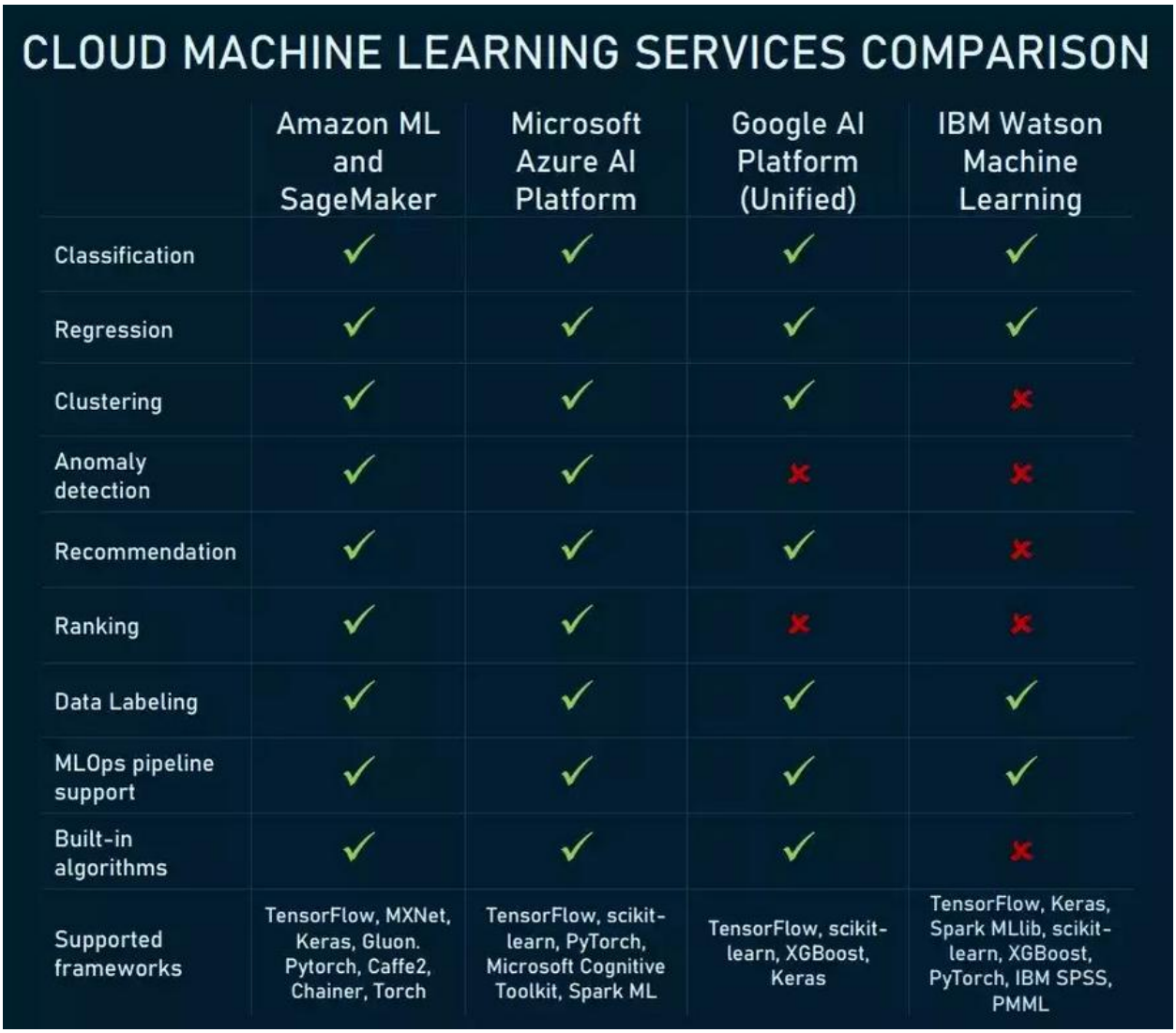
5.1. MLaaS – Машинско учење како услуга

Машинското учење како услуга (MLaaS) е дефиниција за различни платформи базирани на cloud кои ги покриваат повеќето инфраструктурни прашања како што се претходна обработка на податоци, обука на модели и евалуација на модели и предвидувања. Резултатите од предвидувањето може да се поврзат со надворешна инфраструктура, односно внатрешната ИТ инфраструктура на било која компанија преку REST API.

Услугите за машинско учење на Amazon, Azure Machine Learning, Google AI Platform и IBM Watson Machine Learning се четири водечки cloud MLaaS услуги кои овозможуваат брза обука и распоредување на модели.

Понатаму ќе дадеме осврт на некои од сервисите кои овие провајдери ги нудат. Сепак, вреди да се спомне дека во однос на платформите за прилагодено моделирање, сите четири провајдери предлагаат слични производи.

На слика 65 е даден визуелен приказ на достапните сервиси за овие провајдери.



Слика: 65

5.1.1. Amazon Machine Learning и SageMaker

Амазон има два главни производи посветени на машинско учење. Претходната платформа наречена Amazon Machine Learning[21] и SageMaker[30], поновата.

5.1.1.1. Машинско учење на Амазон

Amazon Machine Learning за предвидлива аналитика е едно од најавтоматизираните ML решенија на пазарот и најдобро одговара за операции чувствителни на рокови. Услугата може да вчита податоци од повеќе извори, вклучително и Amazon RDS, Amazon Redshift, CSV датотеки, итн. Сите операции за предпроцесирање на податоци се автоматски. Сервисот самиот препознава кои варијабли се категорички, кои нумерички и не се потпира на корисникот за селекција на метод за понатамошно предпроцесирање на податоците, како намалување на димензии.

Способностите за предвидување на Amazon ML се ограничени на три опции: бинарна класификација, повеќекласна класификација и регресија. Имајќи го ова во предвид, оваа услуга на Amazon ML не поддржува никакви методи за учење без надзор, а корисникот мора да избере целна променлива од множеството за тренирање која би била лабелирана. Исто така, од

корисникот не се бара да знае методи за машинско учење бидејќи Амазон ги избира автоматски откако ќе ги анализира дадените податоци.

Од 2021 година Амазон повеќе не ја ажурира ниту документацијата, ниту самата платформа за машинско учење. Услугата е сè уште функционална, но не прифаќа нови корисници. Тоа е затоа што SageMaker и сите негови соодветни услуги се супериорни во однос на AML и во основа ја испорачуваат истата функционалност на корисниците. Целта на овој потег е да се поттикнат корисниците да се мигрираат на новата и подобрена платформа.

Предиктивната аналитика може да се користи во форма на податоци во реално време или на барање, преку два одделни API кои се достапни на корисниците.

Високото ниво на автоматизација инкорпорирано во услугата делува и како предност и како недостаток за употреба на Amazon ML. Доколку автоматизиран и предефиниран процес одговара на потребите на бизнисот, тогаш оваа платформа може да е адекватна, во спротивно Амазон го нуди SageMaker.

5.1.1.2. SageMaker

SageMaker е платформа за машинско учење што треба да ја поедностави работата на научниците за податоци преку обезбедување алатки за брзо градење и распоредување на модели. На пример, тој обезбедува Jupyter тетратки за пишување, за поедноставување на истражувањето и анализата на податоците без проблеми со управувањето со серверот.

Во 2021 година, Амазон го лансираше SageMaker Studio, првиот IDE за машинско учење. Оваа алатка обезбедува веб-базиран интерфејс кој овозможува извршување на сите тестови за обука на ML модел во една единствена околина. Сите развојни методи и алатки, вклучувајќи тетратки, инструменти за дебагирање, моделирање на податоци и негово автоматско создавање се достапни преку SageMaker Studio.

Амазон, исто така, има вградени алгоритми кои се оптимизирани за големи збирки на податоци и пресметки во дистрибуирани системи. Тие вклучуваат:

- „Линеарен ученик“ е метод за класификација со надзор
- Машини за факторизација, наменети за класификација и регресија дизајнирани за сетови со нецелосни податоци.
- XGBoost е алгоритам за зголемени дрвја со надзор кој ја зголемува точноста на предвидувањето во класификацијата, регресијата и рангирањето со комбинирање на предвидувањата на поедноставните алгоритми.
- Класификацијата на сликите се заснова на ResNet, која исто така може да се примени за учење со трансфер.
- Seq2seq е алгоритам за предвидување секвенци со надзор (на пример, преведување реченици, претворање низи зборови во пократки како резиме, итн.)
- K-means, метод за учење без надзор за кластерирање
- Анализата на главна компонента, се користи за намалување на димензиите.
- Латентна распределба на Дирихле, метод без надзор кој се користи за наоѓање категории во документите.
- Невронски модел на тема (NTM) е метод без надзор кој ги истражува документите, ги открива најзначајните зборови и ги дефинира темите (корисниците не можат однапред да ги дефинираат темите, но можат да го постават очекуваниот број од нив).
- Прогнозирањето DeepAR е надгледуван алгоритам за учење што се користи за предвидување временски серии што користи рекурентни невронски мрежи (RNN).

- BlazingText е алгоритам за обработка на природен јазик (NLP) изграден врз основа на Word2vec, кој му овозможува да мапира зборови во големи збирки текстови со векторски претстави.
- Random Cut Forest е алгоритам без надзор за откривање аномалија, способен да доделува резултати за аномалија на секоја податочна точка.
- Learning to Rank (LTR) е приклучок за Amazon Elasticsearch кој овозможува ML рангирање на барањата за резултатите од пребарувањето.
- K-најблискиот сосед (k-NN) е алгоритам базиран на индекс што може да се користи заедно со невронски модел за тема за да се изградат сопствени услуги за препораки. Исто така, постои посебен мотор за персонализирање на Amazon за препораки во реално време што го користи самиот Amazon.com.

Вградените методи на SageMaker во голема мера се вкрстуваат со ML API што ги предлага Амазон, но тука им овозможува на научниците за податоци да си играат со нив и да ги користат сопствените збирки на податоци.

Доколку горе наведените методи не се соодветни на потребите на корисникот, корисникот може да додаде свои методи и да стартува модели преку SageMaker користејќи ги неговите карактеристики за распоредување. Исто така може да се интегрира SageMaker со TensorFlow, Keras, Gluon, Torch, MXNet и други библиотеки за машинско учење.

Општо земено, услугите за машинско учење на Амазон обезбедуваат доволно слобода и за искусни научници за податоци и за оние на кои им треба само работите да се завршат без да копаат подлабоко во подготовките и моделирањето на бази на податоци. Ова би бил солиден избор за компаниите кои веќе користат cloud услуги на Амазон и не планираат да се префрлат на друг cloud провадер.

Популарноста на DevOps терминот меѓу заедницата за развој на софтвер го дава терминот „MLOps“. DevOps е пристап за развој на софтвер кој сугерира спојување на тимови за развој и развојни операции за да се оптимизираат процесите за развој на софтвер со фокусирање на кратки спринтови и брзи издавања на нови верзии на софтверот. Ова се постигнува со примена на високо ниво на автоматизација на рутинските задачи. MLOps, пак, ги применува истите принципи во машинското учење, што доведе до појава на автоматско управување со податоци, тренинг, распоредување на модели и нивно следење.

Затоа, во 2021 година, провајдерите на MLaaS нудат алатки за практичарите на MLOps да управуваат со овие процеси за машинско учење. Во случајот на Амазон, тие објавија MLOps платформа за градење и управување со MLOps инфраструктура. Доаѓа со шаблони на архитектури што содржи вообичаени услуги AWS за да се овозможи побрз развој на решенија за конкретен бизнис проблем.

5.1.2. Microsoft Azure AI платформа

Azure[3] AI платформата претставува унифицирана платформа за машинско учење со своите API и инфраструктурни услуги. Така, овде ќе ја споменеме секоја голема услуга што ја нуди Azure во однос на решенијата за машинско учење.

5.1.2.1. Azure Machine Learning

Azure Machine Learning е главната средина за управување со податоци, обука за модели и распоредување.

Платформата обезбедува Студио за машинско учење, средина базирана на веб и малку код, за брзо конфигурирање на операциите и процесите за машинско учење. Општо земено, Azure Studio има средства за истражување на податоци, претходна обработка, избор на методи и потврдување на резултатите од моделирањето. Студиото поддржува околу 100 методи кои се однесуваат на класификација (бинарна + повеќекласа), откривање аномалии, регресија, препораки и анализа на текст. Вреди да се спомене дека платформата има еден алгоритам за кластерирање (K-means).

Како и кај Амазон, Azure нуди интеграција со Jupyter што значи дека корисникот може да го напише својот код и директно да го изврши и тестира во ML Studio. Исто така, обезбедува ONNX Runtime за забрзување на ML моделите низ низа ОС и хардверски платформи. Исто така, поддржана е и интероперабилност помеѓу различни МЛ платформи. Azure поддржува повеќе такви платформи како што се TensorFlow, PyTorch[7], scikit-learn и други.

Постојат голем број функции во ML Studio кои вреди да бидат споменати.

- Azure Machine Learning дизајнер е графички интерфејс со „влечење и пуштање“ функционалност за ML студио што обезбедува пристап и контрола врз карактеристиките на платформата. Овде, можете да се менуваат додатоците, да се селектираат ML методи за употреба и да закачат готовите решенија на серверот.
- Automated ML е SDK што обезбедува обука без код за модели со малку код. Во основа, Automated ML го надополнува ML studio со висок степен на автоматизација за рутински задачи и поддршка за истражување на податоци, прилагодување на моделот и закачување на сервер. Azure ги наведува задачите за класификација, регресија и прогнозирање на временски серии како применливи за обука со Automated ML алатката.
- Azure ML Python и R јазичните SDK се целосно интегрирани во ML Studio.
- Поддршка за ML-платформи како PyTorch, TensorFlow и scikit-learn, дополнително, Azure нуди интероперабилност помеѓу овие користејќи ONNX Runtime.
- Вградени се модуларни текови и тоа му овозможува на корисниците да конструираат приспособен тек за податоци соодветен за конкетниот проект за машинско учење.
- Поддршка за проекти за означување на податоци
- Ресурсите за пресмтка се исто така прилагодливи, поддржуваат различни услуги како што се услугите на Azure Kubernetes, контејнери и кластери за пресметување
- Постои алатка MLOps за управување, распоредување и следење на модели во рамките на автоматизираните текови.

И ML Designer и Automated ML обезбедуваат средства за неискусните корисници да изградат ML решенија. Дополнително, студиото за учење Azure Machine вклучува голем број функции што може да ги користат научниците за податоци кои се запознаени со постоечките технологијата и решенија. Но, ова не ги ограничува овие алатки, бидејќи Azure ML е наменета да се користи како единствена платформа со сите свои способности.

Започнување на машинско учење со Azure подразбира одредена крива на учење. Но, тоа на крајот води до подлабоко разбирање на сите главни техники во оваа област. Графичкиот интерфејс на Azure ML го визуелизира секој чекор во работниот тек и ги поддржува почетниците во сферата на машинското учење. Можеби главната придобивка од користењето на Azure е разновидноста на алгоритми достапни за играње.

5.1.2.2. Галерија Azure AI

Друг голем дел од Azure ML е галеријата Azure AI. Тоа е збирка решенија за машинско учење обезбедени од заедницата кои може да бидат истражени и повторно искористени од научниците за податоци. Овој Azure продукт е моќна алатка за започнување со машинско учење и за вовед на новите корисници во услугите понудени од Azure.

5.1.2.3. Azure Percept

Од 2021 година, Azure отвори преглед за Azure Percept. Главната идеја зад Percept е да се обезбеди SDK за создавање модели кои можат да се интегрираат со хардверски уреди партнери на Microsoft. Ова подразбира лесен начин за градење и интегрирање на компјутерска визија или алатки за препознавање говор. Дополнително, постои цела палета на API кои можат да се поврзат со системот.

Ажурирањата на платформата за 2019 година главно се фокусираат на Python Machine Learning SDK и лансирањето на Azure ML Workspaces (во основа интерфејс за ML платформа). Тоа им овозможува на програмерите да распоредуваат модели, да визуелизираат податоци и да работат на подготовка на податоци на едно место.

5.1.3. Google AI платформа (унифицирана)

Google AI Platform[32] (Unified) обединети алатки за ML кои претходно постоеја одделно. Платформата се состои од AI платформа (класична), AutoML, платформи и API под капакот на AI Platform Unified.

5.1.3.1. Платформа за вештачка интелигенција (класична)

AI Platform Classic е алатка која вклучува голем број функции за експерти за машинско учење и научници за податоци. AI Platform Classic ги нуди следните услуги за градење сопствени модели:

- Сервисот за обука обезбедува околина за градење модели, користејќи вградени алгоритми или користејќи кориснички (сопствени) алгоритми. Корисниците можат да поднесат свои методи за обука или да креираат сопствени контејнери за да ја инсталираат апликацијата за обука.
- Предиктивна услуга ви овозможува интеграција на генерираните предвидувања во конкретните бизнис апликации или која било друга услуга.
- Услугата за означување на податоци е алатка која бара човечки тим да го означи множеството на податоци. Услугата поддржува етикетирање за видео, текст и слики кои ќе бидат обработени по кориснички упатства.
- Deep Learning Image обезбедува спецификација за виртуелна машина за длабоко учење. Сликата е претходно конфигурирана за задачи за ML и наука за податоци со претходно инсталирани популарни рамки и алатки.
- Тетратките на платформата за вештачка интелигенција се местото каде што корисникот може да креира/управува со примери на виртуелни машини и да ги конфигурира типовите меморија за обработка на податоци (процесор или графички процесор). Доаѓа и претходно интегриран со примери на TensorFlow и PyTorch, пакети за длабоко учење и тетратка Jupyter.

Управувањето со моделите, работните места и крајните точки може да се врши преку посветен REST API, командна линија gcloud или Google Cloud Console.

AI Platform Classic е дизајниран за искусни корисници.

5.1.3.2. Google Cloud AutoML

Google Cloud AutoML е ML платформа базирана на cloud, која овозможува пристап без код за градење решенија управувани од податоци. AutoML е дизајниран за градење сопствени модели и за неискусните но и за искусни инженери за машинско учење. Но, платформата предлага и колекција на претходно изградени модели достапни преку сет на API.

Главниот концепт на платформата на Google е опишан преку нејзините градежни блокови за вештачка интелигенција. Овие се суштински различни алатки како AutoML, TensorFlow и API кои треба да се користат заедно за да се изградат решенија за ML. Ова значи дека корисникот може да комбинира сопствен модел и претходно обучени модели во еден производ.

Понатаму, решенијата за ML може да се распоредат на корисничка веб-локација или на посебна инфраструктура за вештачка интелигенција која вклучува различни методи за обработка на податоци на графичкиот процесор или основниот процесор. Непотребно е да се каже дека AutoML е целосно интегриран со сите услуги на Google и складира податоци во облакот. Обучените модели може да се распоредат преку REST API интерфејсот.

Во поглед на платформата како целина, постојат два вида решенија кои се наменети да ги користат различни корисници. Платформата за вештачка интелигенција (Класик) обезбедува повеќе опции за создавање сопствени модели и рачно управување со алгоритми и процеси за обука. И најдобро одговара за искусни развивачи на машинско учење. Спротивно на тоа, AutoML овозможува без код градење на модели, аплицирање на податоци и интегрирање предвидувања секаде каде што е потребно.

5.1.4. TensorFlow

TensorFlow[12] е уште еден производ на Google, кој е библиотека за машинско учење со отворен код која нуди различни алатки за работа со податоци наместо ML-како-услуга. Нема визуелен интерфејс и кривата на учење за TensorFlow би била доста тешка. Сепак, библиотеката е насочена и кон софтверски инженери кои планираат транзиција кон науката за податоци. Google TensorFlow е доста моќен, но главно насочен кон задачи на длабоки невронски мрежи.

Во основа, комбинацијата на услугата TensorFlow и Google Cloud предлага решенија за инфраструктура-како-услуга и платформа-како-услуга според тристепениот модел на cloud услуги.

Решението MLOps од Google нуди способности слични на оние од AWS за изградба и управување со процедурални текови за машинско учење. Но, бидејќи Azure предлага модуларен систем претходно конфигуриран за употреба во ML Studio, нивното решение изгледа супериорно меѓу овие три продавачи.

5.1.5. IBM Watson Machine Learning Studio

Платформата IBM Machine Learning е организирана како и претходните провајдери. Технички, системот нуди два пристапи: автоматизиран и рачен - за стручни практичари.

5.1.5.1. Watson Studio u AutoAI

Watson Studio[22] има AutoAI што носи целосно автоматизирана обработка на податоци и интерфејс за градење модели на кој му треба малку или воопшто не му е потреба никаква обука за да започне со обработка на податоци, подготовка на модели и нивно распоредување во производство.

Автоматизираниот дел може да реши три главни типа на задачи: бинарна класификација, повеќекласна класификација и регресија. Може да се изберете или целосно автоматизиран

пристап или рачно да се избере методот ML што ќе се користи. Во моментов, IBM има десет методи за покривање на овие три групи на задачи:

- Логистичка регресија
- Класификатор на дрво на одлуки
- Случаен шумски класификатор
- Класификатор на дрво засилен со градиент
- Наивна Баесова
- Линеарна регресија
- Регресор на дрвото на одлуки
- Случаен шумски регресор
- Градиент засилен регресор на дрво
- Изотонична регресија

Покрај AutoAI, има уште две услуги што можете да се искористат за градење модели.

5.1.5.2. SPSS моделер

SPSS е софтверски пакет кој се користи за трансформирање на податоците во статистички деловни информации. Преземан од IBM во 2009 година и интегриран како самостојна ML услуга, сега е производ без графички кориснички интерфејс кој овозможува да се прикачуви множеството податоци, да се користат SQL изјави за манипулирање со податоци и обучување модели за работа со деловни информации.

5.1.5.3. Невронска мрежа и длабоко учење

Услугата за невронски мрежи и длабоко учење е малку поинаква од SPSS Modeler. Тоа е алатка за моделирање на невронски мрежи преку посветен GUI. Услугата е интегрирана во Watson Studio што овозможува управување со податоци со неговата вградена алатка за интеграција на податоци. Главниот фокус на услугата се способностите за длабоко учење и обуката со големи множестава податоци. Дополнително, услугите на невронската мрежа се интегрирани со еден куп ML платформи како Keras, PyTorch или TensorFlow.

Одделно, IBM нуди работен тек на обука за длабока невронска мрежа со интерфејс за уредувач на проток сличен на оној што се користи во Azure ML Studio.

За понапредни функционалности IBM ML поддржува Јупитер за рачно програмирање на модели користејќи популарни платформи како TensorFlow, scikit-learn, PyTorch и други.

Кога зборуваме за платформите за машинско учење како услуга (MLaaS), се чини дека Azure моментално ја има најразновидната група алатки на пазарот MLaaS. Го покрива најголемиот дел од задачите поврзани со ML, обезбедува два различни производи за градење сопствени модели и има солиден сет на API за оние кои не сакаат да ја напаѓаат науката за податоци со голи раце.

Едно од најновите ажурирања направени во 2019 година е прекинот на производството на стариот модел, кој беше заменет со AutoAI. Моделите обучени со стариот модел сè уште можат да се користат во ML Studio, но новите модели сега може да се обучуваат во AutoAI. Другите ажурирања се однесуваат на поддршката за најновите верзии на TensorFlow и Python.

5.1.6. Споредба на API за машинско учење на Amazon, Microsoft, Google и IBM

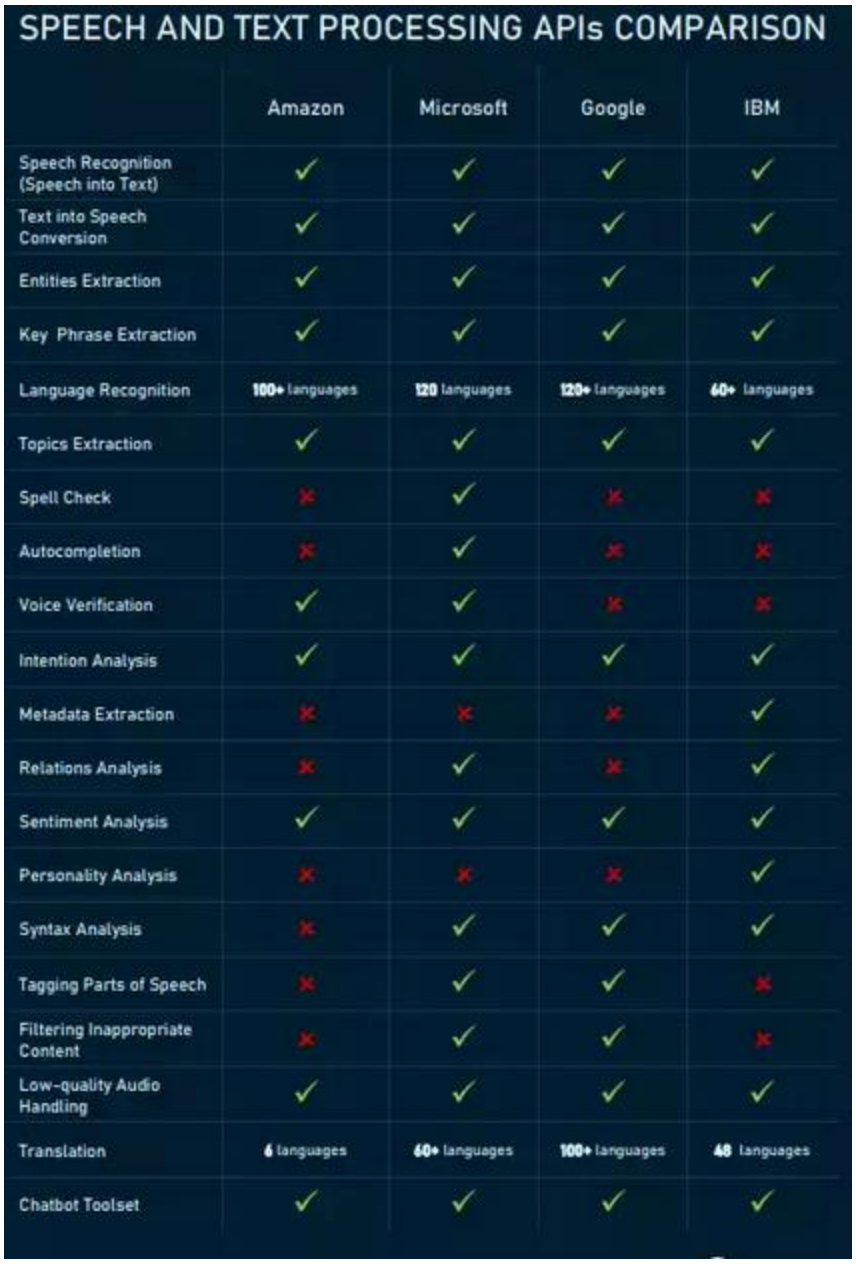
Покрај целосните платформи, достапни за корисниците се и API на високо ниво. Ова се услугите со веќе обучени модели во кои корисникот може да ги додаде своите податоци и да добие резултати. API-ата воопшто не бараат експертиза за машинско учење. Во моментов, API-ата од овие четири продавачи може да се поделат во три големи групи:

- препознавање на текст, превод и анализа на текстот

- препознавање на слики + видео и поврзана анализа

- друго, што вклучува специфични некатегоризирани услуги

Мајкрософт ја нуди најбогатата листа на функции. Сепак, најосновните и најважните се достапни од сите продавачи. На слика 66 е дадена споредбата на API-ата за текст и говор.



Слика: 66

5.1.6.1. API за обработка на говор и текст: Amazon

Амазон обезбедува повеќе API-а кои имаат за цел популарни задачи во рамките на анализата на текстот. Овие се исто така високо автоматизирани во однос на машинското учење и само им е потребна соодветна интеграција за да работат.

- Амазон Лекс. Lex API е создаден за вградување чет-ботови во вашите апликации бидејќи содржи капацитети за автоматско препознавање говор (ASR) и обработка на природен јазик (NLP). Тие се засноваат на модели за длабоко учење. API-то може да препознае пишан и изговорен текст, а интерфејсот Lex овозможува да се закачат препознатливите влезови на различни решенија на backend. Очигледно, Амазон ја охрабрува употребата на својата Lambda cloud околина. Затоа, пред да се искористи претплата на Lex, препорачливо

е прво запознавање со Lambda. Покрај самостојните апликации, Lex моментално поддржува распоредување чет-ботови за Facebook Messenger, Slack и Twilio.

- Транскрипција на Амазон. Додека Lex е сложена алатка ориентирана кон chatbot, Транскрипција е создадена исклучиво за препознавање на говорен текст. Алатката може да препознае повеќе говорници и работи со звук од телефонија со низок квалитет. Ова го прави API-то практично решение за каталогизирање аудио архиви или добра поддршка за понатамошна анализа на текст на податоците од центар за повици.
- Амазон Поли. Услугата Polly е нешто обратна од Lex. Го претвора текстот во говор, што ќе им овозможи на чет-ботовите да одговорат со глас. Сепак, нема да го состави текстот, само направи текстот да звучи блиску до човечки говор. Во моментов, поддржува женски и машки гласови за 30 јазици, главно англиски и западноевропски. Некои јазици имаат повеќе женски и машки гласови, така што има дури и разновидност за избор. Како Lex, Polly се препорачува за употреба со Lambda.
- Amazon Comprehend. Comprehend е уште еден NLP сет на API кои, за разлика од Lex и Transcrib, имаат за цел различни задачи за анализа на текст. Во моментов, Comprehend поддржува:
    - Екстракција на ентитети (препознавање имиња, датуми, организации, итн.)
    - Откривање на клучни фрази
    - Препознавање јазик
    - Анализа на чувства (колку е позитивен, неутрален или негативен текстот)
    - Моделирање на теми (дефинирање на доминантни теми преку анализа на клучни зборови)

  Оваа услуга може да се користи за анализа на одговорите на социјалните мрежи, коментари и другите големи текстуални податоци кои не се подложни на рачна анализа. На пример: комбинацијата на „Сфати“ и „Транскрипција“ може да помогне да се анализираат чувствата во услугата за клиентите управувана од телефонијата.
- Амазон превод. Како што наведува името, услугата „превод“ преведува текстови. Амазон тврди дека користи невронски мрежи кои - во споредба со пристапите за превод базирани на правила - обезбедуваат подобар квалитет на преводот. За жал, сегашната верзија поддржува превод од само шест јазици на англиски и од англиски на тие шест. Јазиците се арапски, кинески, француски, германски, португалски и шпански.

5.1.6.2. API за обработка на говор и текст: Когнитивни услуги на Microsoft Azure

Исто како Амазон, Мајкрософт нуди API на високо ниво, когнитивни услуги, кои можат да се интегрираат со компаниската инфраструктура и да извршуваат задачи без потреба од експертиза во науката за податоци.

- Говор. Сетот за говор содржи четири API кои применуваат различни типови на техники за обработка на природен јазик (NLP) за природно препознавање говор и други операции:
    - Преведувач говор API
    - Bing Speech API за претворање на текст во говор и говор во текст
    - API за препознавање звучници за задачи за гласовна потврда
    - Прилагодена услуга за говор за примена на капацитетите на Azure NLP користејќи сопствени податоци и модели
- Јазик. Јазичната група на API се фокусира на текстуална анализа слична на AmazonComprehend:

- Интелигентната услуга за разбирање јазици (LUIS) е API што ги анализира намерите во текстот што треба да се препознаат како команди (на пр. „изврши ја апликацијата YouTube“ или „вклучи ги светлата во дневната соба“)
- API за анализа на текст за анализа на чувствата и дефинирање на теми
- Проверка на правопис на Бинг
- Преведувач текст API
- Web Language Model API што ги проценува веројатностите на комбинации на зборови и поддржува автоматско пополнување на зборови
- API за лингвистичка анализа што се користи за одвојување реченици, означување на делови од говорот и делење текстови во означени фрази

5.1.6.3. API за обработка на говор и текст: Услуги на Google Cloud ML/ Cloud AutoML

Иако овој сет на API главно се вкрстува со она што го предлагаат Amazon и Microsoft Azure, има некои интересни и уникатни работи што треба да се погледнат. Бидејќи се појави платформата AutoML наместо Prediction API, сега таа ги проширува можностите на услугите на Google Cloud ML. Значи, секое API за автоматско машинско учење од Google е вистинска опција за обука на сопствени модели.

- Dialogflow - Со различни чет-ботови кои се на врвот на денешните трендови, Google исто така има што да понуди. Dialogflow се напојува со NLP технологии и има за цел да ги дефинира намерите во текстот и да толкува што сака личноста. API може да се подесува и прилагоди за потребните намери користејќи Java, Node.js и Python.
    - API за природен јазик. Овој е речиси идентичен во неговите основни карактеристики со Comprehend by Amazon и Language by Microsoft.
    - Дефинирање на ентитети во текстот
    - Препознавање на чувства
    - Анализирање на синтаксните структури
    - Категоризирање на теми (на пр. храна, вести, електроника, итн.)
- Speech-to-Text API Оваа услуга го препознава природниот говор, а можеби нејзината главна придобивка во споредба со слични API е изобилството на јазици поддржани од Google. Во моментов, неговиот речник работи со над 125 глобални јазици и нивни варијанти. Исто така, има некои дополнителни функции:
    - Советите за зборови овозможуваат прилагодување на препознавањето на специфични контексти и зборови што може да се изговорат (на пр. за подобро разбирање на локалниот или индустрискиот жаргон)
    - Филтрирање на несоодветна содржина
    - Ракување со бучно аудио
- Cloud translation API. Во основа, ова API може да се користи за интергација на Google Translate во кориснички производи. Овој вклучува над сто јазици и автоматско откривање јазици.
- AutoML NL API. Во основа, она што го прави е дозволува да ги прикачување на податоци за обука преку AutoML UI и да тренирање на сопствени модели. Достапни карактеристики се:
    - Дефинирање на содржината на англиски јазик
    - Дефинирање на ентитети во текстот
    - Анализирање на структурата на синтаксата
- AutoML API за превод. Дозволува креирање на кориснички специфични модели за превод. Намената е за подобар превод на текстови од специфичен домен.

5.1.6.4. API за обработка на говор и текст: IBM Watson

IBM се натпреварува и за пазарот на API. Следи преглед на API-ата кои ги нуди.

- Говор во текст. Во моментов, IBM нуди препознавање говор за 9 јазици, вклучувајќи јапонски, португалски и арапски. API-то може да препознае повеќе говорници, да забележи клучни зборови и да ракува со аудио со загуби. Интересна карактеристика е фаќањето алтернативи на зборови и нивното известување. На пример, ако системот го забележи зборот „Бостон“, може да претпостави дека може да има алтернатива „Остин“. По анализата на неговата хипотеза, API доделува оценка за доверба на секоја алтернатива.
- Текст во говор. Интересно е што 9 јазици од текст во говор само делумно се совпаѓаат со оние во API од говор во текст. Додека двата производи поддржуваат западноевропски јазици, на текст во говор му недостасуваат корејски и кинески. Англискиот, германскиот и шпанскиот дозволуваат избор помеѓу машки и женски гласови; останатите јазици доаѓаат само со женски гласови. Ова е во согласност со трендот гласовните асистенти да имаат женски гласови.
- Јазичен преведувач. Ова API поддржува 48 јазици за превод од и на англиски. Дополнително, може да се додаваат сопствени модели и да се прошири јазичната покриеност. Во моментов, APIто за преведување е пренапишано во неколку посебни услуги со сопствен ценовен модел.
- Класификатор на природен јазик. За разлика од повеќето споменати API, класификаторот на IBM не може да се користи без корисничка сопствена база на податоци. Во основа, алатката овозможува обука на модели користејќи ги корисничките сопствени деловни податоци и потоа овозможува ги класификација на дојдовните записи. Вообичаени случаи на употреба се означување производи во е-трговија, откривање измами, категоризирање пораки и слично.
- Разбирање на природен јазик. Функцијата за разбирање јазик поставена во IBM е доста обемна. Покрај стандардното извлекување информации како екстракција на клучни зборови и ентитети со синтаксна анализа, API нуди голем број интересни способности кои не се достапни од други провајдери. Тие вклучуваат анализа на метаподатоци и пронаоѓање на односи меѓу ентитетите. Дополнително, IBM нуди посебна средина за обука на сопствени модели за анализа на текст со помош на Knowledge Studio.
- Увид на личноста. Релативно невообичаено API. Овозможува анализа на текстови и извлекување индиции за тоа каков однос кон светот има писателот. Ова во основа значи дека системот ќе се врати:
    - карактеристики на личноста (на пр. согласност, совесност, екстравертност, емоционален опсег и отвореност)
    - потреби (на пр. љубопитност, возбуда, предизвик)
    - вредности (на пр. помагање на другите, постигнување успех, хедонизам).

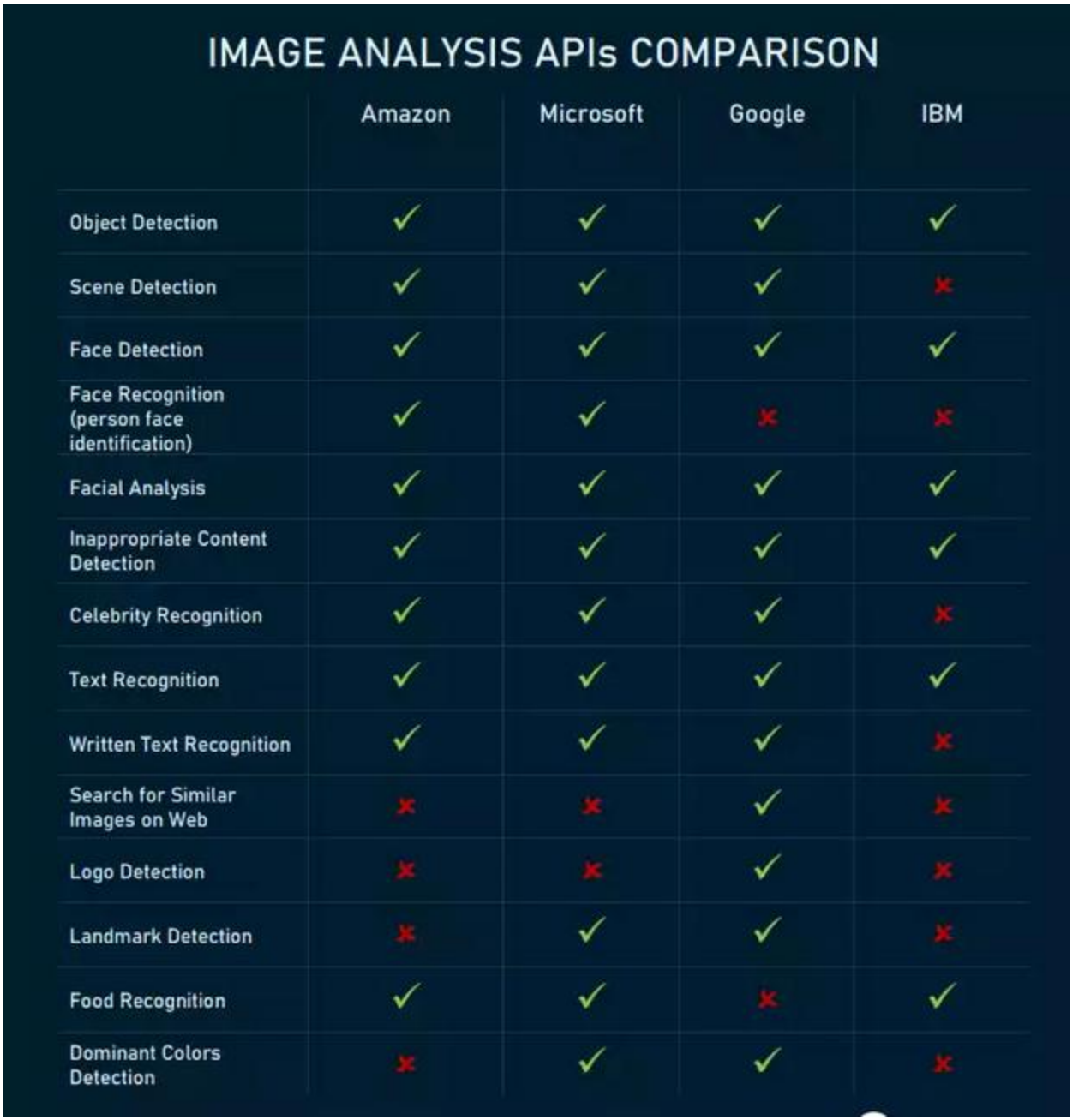
  Врз основа на овие податоци, API-то може да одреди потрошувачки преференици (на пр. музика, учење, филмови). Најчестиот случај за употреба на таков систем е анализа на содржина генерирана од корисникот за прецизен маркетинг на производи.
  Важно е да се напомене дека Personality Insights беше прекината и ќе ги поддржува постоечките примероци што го користат APIто до крајот на 2021 година.
- Тонски анализатор. Анализаторот на тонови е посебен API кој се фокусира на анализа на чувствата и е насочен кон истражување на социјалните медиуми и разни аналитики за ангажирање на клиентите. Анализаторот покрива само пишан текст и не извлекува увид од говорните формати.

Покрај текстот и говорот, Amazon, Microsoft, Google и IBM обезбедуваат прилично разновидни API за анализа на слики и видео. Најразновидните алатки за анализа на слики моментално се достапни на Google Cloud. Додека анализата на слики тесно се вкрстува со видео API, многу алатки за видео анализа сè уште се во развој или бета верзии. На пример, Google нуди поддршка за различни задачи за обработка на слики, но нема функции за видео анализа што веќе се достапни во Microsoft и Amazon.

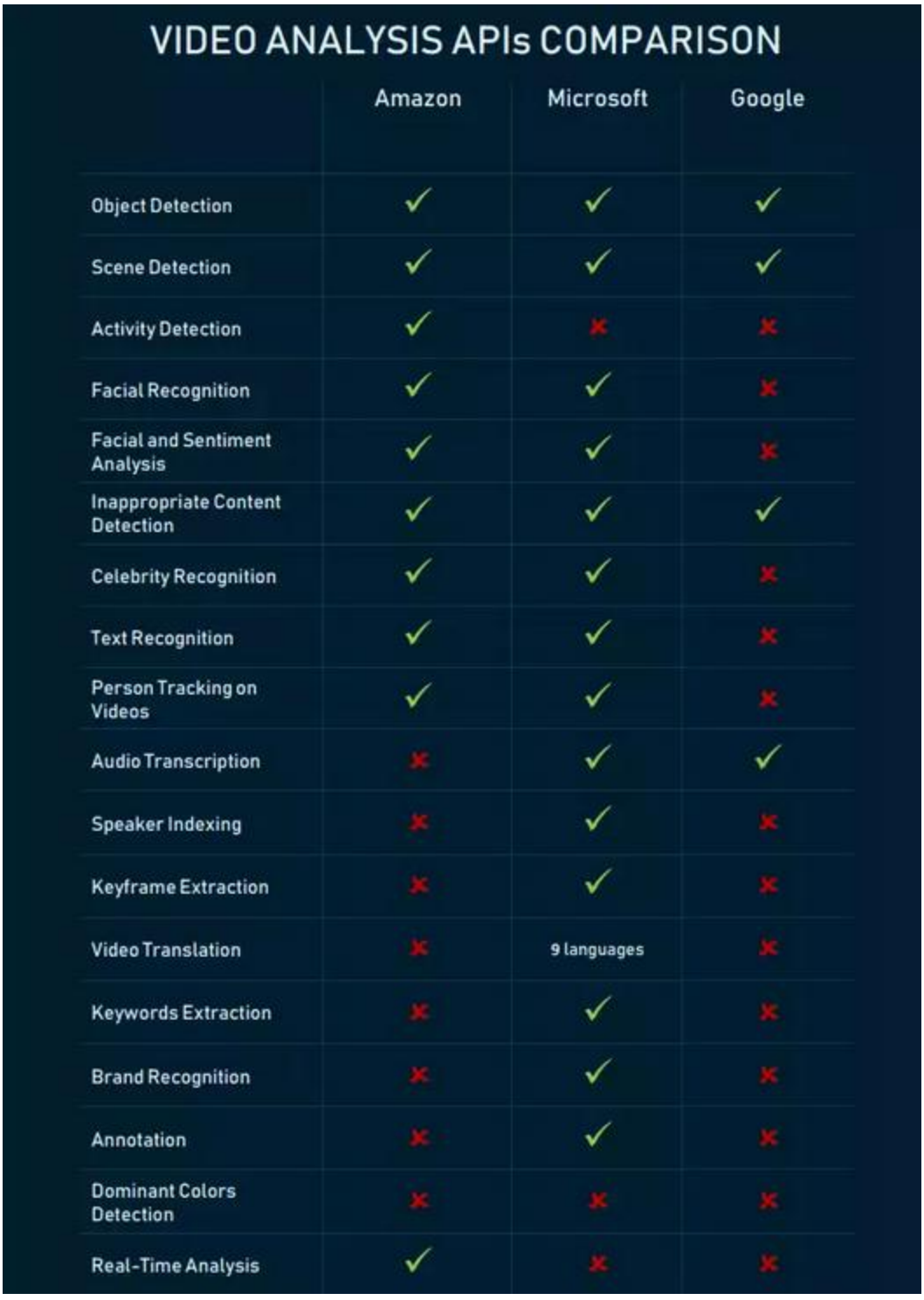
5.1.7. API за машинско учење за видео анализа од Amazon, Microsoft, Google и IBM

Мајкрософт изгледа како победник, иако сè уште можеби Амазон ги има најефикасните API-а за анализа на видео бидејќи поддржува стриминг видеа. Оваа функција опипливо го проширува спектарот на случаи на употреба. IBM не поддржува API за видео анализа.

На слика 67 е дадена табеларна споредба на достапните API-а за анализа на слика. На слика 68 е дадена слична табеларна споредба на API-ата за обработка на видео записи.


IMAGE ANALYSIS APIs COMPARISON
Amazon
Microsoft
Google
IBM
Object Detection
Scene Detection
Face Detection
Face Recognition (person face identification)
Facial Analysis
Inappropriate Content Detection
Celebrity Recognition
Text Recognition
Written Text Recognition
Search for Similar Images on Web
Logo Detection
Landmark Detection
Food Recognition
Dominant Colors Detection

Слика: 67



Слика: 68

5.1.7.1. API за обработка на слики и видео: препознавање на Amazon

Rekognition API[38] се користи за задачи за препознавање слики и, неодамна, видео. Тие вклучуваат:

- Откривање и класификација на предмети (пронаоѓање различни објекти на сликите и дефиниција на истите)
- Во видеата, може да открие активности како „танцување“ или сложени дејства како „гасне пожар“
- Препознавање на лица (за откривање лица и пронаоѓање соодветни) и анализа на лицето (ова е прилично интересно бидејќи открива насмевки, анализира очи, па дури и дефинира емоционално чувство во видеата)
- Откривање на несоодветни видеа
- Препознавање на познати личности во слики и видеа

5.1.7.2. API за обработка на слики и видео: Когнитивни услуги на Microsoft Azure

Пакетот Vision од Microsoft комбинира шест API-а кои се фокусираат на различни типови на анализа на слики, видео и текст.

- Компјутерска визија која препознава предмети, дејства (на пр. одење), напишани и отчукувани текстови и дефинира доминантни бои на сликите
- Модераторот на содржина открива несоодветна содржина во слики, текстови и видеа
- Face API открива лица, ги групира, дефинира возраст, емоции, пол, пози, насмевки и влакна на лицето
- Emotion API е уште една алатка за препознавање лица која ги опишува изразите на лицето
- Custom Vision Service поддржува градење сопствени модели за препознавање слики користејќи кориснички сопствени податоци
- Индексер на видео е алатка за наоѓање луѓе во видеата, дефинирање на чувството на говор и обележување клучни зборови.

5.1.7.3. API за обработка на слики и видео: Услуги на облак на Google/ Cloud AutoML

Cloud vision AI. Алатката изградена за задачи за препознавање слики и е доста моќна за наоѓање специфични атрибути на слики:

- Обележување предмети
- Откривање лица и анализа на изрази (без конкретно препознавање или идентификација на лице)
- Наоѓање знаменитости и опишување на сцената (на пр. одмор, свадба, итн.)
- Пронаоѓање текстови во слики и идентификување јазици
- Доминантни бои

Cloud vision AI. API за препознавање видео од Google кое ја обезбедува следната група алатки:

- Означување на објекти и дефинирање на дејства
- Идентификување на експлицитна содржина
- Препишување говор

AutoML Vision API. AutoML, исто така, излезе со неколку производи за обука на модели, а AutoML Vision е првиот објавен. Функционалности кои ги нуди се:

- Обележување објекти и ангажирање на услуга за етикетирање на луѓе
- Регистрирање обучени модели во AutoML

AutoML видео интелигенција класификација API. Ова е API за обработка на видео, кое може да класифицира одредени снимки од корисничко видео користејќи кориснички сопствени ознаки за податоците.

Додека на ниво на список на функции, на услугите на Google AI можеби им недостасуваат некои способности, моќта на Google API е во огромните збирки на податоци до кои Google има пристап.

5.1.8. API за обработка на слика и (не) видео: Визуелно препознавање на IBM

API-то за визуелно препознавање на IBM моментално не поддржува видео анализа, која е веќе достапна кај други провајдери. И системот за препознавање слика нуди основен сет на функции, малку ограничени во споредба со она што го нудат другите продавачи:

- Препознавање на објекти
- Препознавање лица (API-то враќа возраст и пол)
- Препознавање храна
- Откривање на несоодветна содржина
- Препознавање текст

5.1.9. Специфични API и алатки

Cloud сервис провајдерите нудат множества алатки кои не спаѓаат во ниту една од веќе споменатите категории. Следи краток осврт кон нив.

5.1.9.1. Azure Bot Service

Мајкрософт вложи многу напор да им обезбеди на своите корисници флексибилен пакет алатки за развој на ботови. Во основа, услугата содржи целосна околина за градење, тестирање и распоредување на ботови со користење на различни програмски јазици.

Интересно, услугата бот не бара нужно пристапи за машинско учење. Бидејќи Мајкрософт обезбедува пет шаблони за ботови (основни, форми, разбирање јазик, проактивни и прашања и одговори), само типот за разбирање јазик бара напредни техники за вештачка интелигенција.

Може се користат технологиите .NET и Node.js за да изградба на ботови со Azure и да се интегрираат со повеќе платформи, како:

- Bing
- Cortana
- Skype
- Web Talk
- Е-пошта на Office 365
- GroupMe
- Facebook Messenger
- Slack
- Kik
- Telegram
- Twilio

5.1.9.2. AWS ML Хардвер

Неодамна воведените физички производи од Amazon се преполни со посветени API-а за програмирање на хардвер со модели за длабоко/машинско учење. Производите базирани на ML-алгоритам на Amazon се поделени во три единици:

- AWS DeepLens е програмабилна камера која се користи за примена на ML на вистинскиот хардвер. Во овој случај, можете да се применат услугите на Amazon ML користејќи ја

камерата што може да се користи за визуелно препознавање на податоци и обука на ML модели на нив.
- AWS DeepRacer е уште еден хардверски дел од ML пакетот кој во основа е 1/18 радио контролиран автомобил кој користи учење со засилување.
- AWS Inferentia е чип прилагоден за обработка на длабоко учење и може да се користи за намалување на трошоците за пресметување. Поддржува TensorFlow, PyTorch и Apache MXNet.

5.1.9.3. Bing Search од Microsoft

Мајкрософт нуди седум API-а кои се поврзуваат со основните функции за пребарување на Bing, вклучувајќи автоматска сугестија, вести, слики и видео пребарување.

5.1.9.4. Knowledge од Microsoft

Оваа група API комбинира анализа на текст со широк спектар на уникатни задачи:

- API за препораки кое овозможува градење на системи за препораки за персонализација на купувањето
- Услугата за истражување на знаење овозможува пишување природни прашања за читање податоците од базите на податоци, визуелизација на податоците и автоматско пополнување прашања
- Entity Linking Intelligence API е дизајниран да ги потенцира имињата и фразите што означуваат соодветни ентитети (на пр. Age of Exploration) и да обезбеди појаснување
- Academic Knowledge API прави автоматско пополнување зборови, наоѓа сличности во документите и во зборови и концепти и бара шаблони на графици во документите
- QnA Maker API може да се користи за усогласување на варијации на прашања со одговори за да се создадат чат-ботови и апликации за грижа за клиентите
- Custom Decision Service е алатка за учење за персонализирање и рангирање на различни типови содржини (на пр. линкови, реклами, итн.) во зависност од преференциите на корисникот

5.1.9.5. Google Cloud Talent

За разлика од конвенционалните пребарувачи за работа кои се потпираат на прецизно совпаѓање на клучни зборови, Google користи машинско учење за да најде релевантни врски помеѓу многу описи на работни места и да избегне двосмисленост. На пример, се стреми да ги намали ирелевантните или премногу широки резултати, како враќање на сите работни места со клучниот збор „асистент“ за барањето „помошник за продажба. Главните функционалности на API-то се:

- Поправање на правописни грешки во пребарувањата при барање работа
- Усогласување на посакуваното ниво на искуство
- Наоѓање релевантни работни места кои може да имаат вклучени доменски изрази и индустриски жаргон (на пр. враќање на „barista“ за барањето „сервер“ наместо „мрежен специјалист“ или „специјалист за ангажман“ за барањето „biz dev“)
- Справување со акроними (на пр. враќање на „асистент за човечки ресурси“ за барањето „HR“)
- Справување со различните описи на една локација

5.1.9.6. Вотсон асистент

Платформата за разговор во Ватсон (поранешен Conversation) е прилично позната меѓу инженерите за вештачка интелигенција кои се специјализирани за разговорни интерфејси. IBM обезбедува целосна инфраструктура за градење и имплементирање на ботови способни за разговори во живо, кои може да користат анализа на ентитети и анализа на корисничките намери во пораките.

Инженерите можат или да користат вградена поддршка на Facebook Messenger и Slack или да создадат клиентска апликација за да го стартува ботот таму.

Досега беше даден осврт на сервисите од четирите најголеми и најпознати светски компании. Но исто така, постојат голем број други решенија за ML-as-a-Service кои доаѓаат од стартап компании и се почитувани од научниците за податоци, како што се PredicSis и BigML.

6. Демо

Од досега кажаното, во доменот на NLP голема ефикасност покажуваат моделите базирани на трансформери. Некои од нив се open source, односно достапни се за јавноста комплетно, архитектура, код. Со ова се поттикнува заедницата да работи, да експериментира со ВИ без да мора секогаш да почнува од почеток. Како последица на тоа, ќе има забрзан развој и надградба на веќе постоечкото.

Hugging face е платформа за податочни науки во најширокатa смисла на зборот. Првенствено се става акцент на заедницата. Секој може да се регистрира на платформата, да креира свое репозитори, да работи и да го направи својот труд достапен за останатите членови на заедницата. Истовремено и да има пристап до работата на останатите членови на заедницата. На Huggingface[5] има огромен број репозиториа со секакви модели кои лесно може да се употребат и надградат.

Моделите достапни на huggingface се поделени според намена, така што може да се најдат модели за:

NLP

- Превод
- Класификација на токени
- Сличност на реченици
- Одговарање на прашања
- Сумаризација на текст
- Генерирање на текст

Копмјутерска визија:

- Класификација на слики
- Сегментација на слики
- Генерирање на слики
- Детекција на објекти

Аудио

- Автоматско препознавање на говор
- Аудио класификација
- Текст во говор
- Детекција на говорна активност

За демонстрација ќе пробаме да искористиме пред-трениран модел од областа на NLP за одговарање на прашања. Кодот го пишуваме на Colab[1]. Colab е продукт од Google Research кој овозможува пишување и извршување на Python код директно во интернет пребарувач без потреба од никакви инсталации. Технички тоа е cloud хостиран Jupyter[2] notebook сервис кој нуди пресметковни ресурси за изнајмување и е целосно бесплатен. Пресметковните ресурси се разбира се ограничени и не е погоден за тренирање на големи модели. Наменет е за користење од страна на ученици и истражувачи за брзо експериментирање и споделување код.

Креираме нов документ во Colab и најпрво што треба да направиме е да ја инсталираме библиотеката која го содржи пред-трениранот модел. За инсталација користиме pip[31] installer. Излезот е даден на слика 69.

```
[5]  pip install transformers

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: transformers in /usr/local/lib/python3.7/dist-packages (4.21.2)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-packages (from transformers) (1.21.6)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/dist-packages (from transformers) (21.3)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.7/dist-packages (from transformers) (6.0)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from transformers) (2.23.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.1.0 in /usr/local/lib/python3.7/dist-packages (from transformers) (0.9.1)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-packages (from transformers) (4.64.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-packages (from transformers) (3.8.0)
Requirement already satisfied: tokenizers!=0.11.3,<0.13,>=0.11.1 in /usr/local/lib/python3.7/dist-packages (from transformers) (0.12.1)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.7/dist-packages (from transformers) (2022.6.2)
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from transformers) (4.12.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.7/dist-packages (from huggingface-hub<1.0,>=0.1.0->transformers) (
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packaging>=20.0->transformers) (3.0.9)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata->transformers) (3.8.1)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->transformers) (2022.6.15)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->transformers) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->transformers) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests->transformers) (1.25.1
```

Слика: 69

Следен чекор е иницијализацијата на моделот за специфичната намена

```
from transformers import pipeline
question_answerer = pipeline("question-answering", model='distilbert-base-cased-distilled-squad')
```

Слика: 70

Како што се гледа на слика 70 иницијализираме distilBert за одговарање прашања.

Следен чекор е дефиниција на контекст. За пример контекст ќе ја земеме приказната за „Пепелашка“, англиски превод од оригиналната верзија од браќата Грим. На слика 71 е прикажана соодветната келија со исечок од текстот.

```
[9]  context = r"""
       Cinderella
     The wife of a rich man fell sick, and as she felt that her end
     was drawing near, she called her only daughter to her bedside and
     said, dear child, be good and pious, and then the
     good God will always protect you, and I will look down on you
     from heaven and be near you.  Thereupon she closed her eyes and
     departed.  Every day the maiden went out to her mother's grave,
     and wept, and she remained pious and good.  When winter came
     the snow spread a white sheet over the grave, and by the time the
     spring sun had drawn it off again, the man had taken another wife.
     The woman had brought with her into the house two daughters,
     who were beautiful and fair of face, but vile and black of heart.
     Now began a bad time for the poor step-child.  Is the stupid goose
     to sit in the parlor with us, they said.  He who wants to eat bread
     must earn it.  Out with the kitchen-wench.  They took her pretty
     clothes away from her, put an old grey bedgown on her, and gave
     her wooden shoes.  Just look at the proud princess, how decked
     out she is, they cried, and laughed, and led her into the kitchen.
     There she had to do hard work from morning till night, get up
     before daybreak, carry water, light fires, cook and wash.  Besides
```

Слика: 71

И тоа е се што е потребно за иницијализација на пред-трениран DistilBERT модел за одговарања прашања. Следно што ни преостанува е да ги поставиме прашањата и да ја анализираме ефикасноста.

Прво прашање: Каде е мајката на Пепелашка? На слика 72 е даден резултатот од прашањето.

```
[10]
     result = question_answerer(question="Where is cindarella's mother?",     context=context)
     print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {result['start']}, end: {result['end']}")

     Answer: 'grave', score: 0.6834, start: 2388, end: 2393
```

Слика: 72

Одговорот е „гроб“ што е точен одговор. Довербата во одговорот е 0,6 што е прилично убедливо и индексите за почеток и крај покажуваат каде во текстот е најден одговорот.

На слика 73 е дадено второто прашање: Колку сестри има Пепелашка?

```
[13] result = question_answerer(question="How many sisters does cindarella have?", context=context)
     print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {result['start']}, end: {result['end']}")

     Answer: 'two', score: 0.8055, start: 13544, end: 13547
```

Слика: 73

Одговорот е точен: две, со доверба од 0,8.

Но што ако поставиме не толку јасно прашање? На слика 74 е даден одговорот на прашањето: каде е Пепелашка?

```
[11] result = question_answerer(question="Where is cindarella?", context=context)
     print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {result['start']}, end: {result['end']}")

     Answer: 'hazel-tree', score: 0.1682, start: 7857, end: 7867
```

Слика: 74

Довербата во одговорот е 0,17 што е релативно добро: моделот не може да знае каде е Пепелашка во моментов бидејќи не е наведено во текстот, но знае дека во еден момент таа била под лешниковото дрво и тоа го враќа како одговор.

На слика 75 е даден одговорот на прашањето: кој пол е Пепелашка?

```
[14] result = question_answerer(question="What gender is cindarella?",     context=context)
     print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {result['start']}, end: {result['end']}")

     Answer: 'foreign princess', score: 0.0396, start: 6341, end: 6357
```

Слика: 75

„Странска принцеза“ со доверба од 0,03. Моделите не можат да размислуваат надвор од дадениот контекст без разлика што се тренирани на гига бајти податоци. Во дадениот текст никогаш не е спомнат полот на Пепелашка и иако на човек му изгледа сосема очигледен одговорот, ако одговорот не постои во контекстот, одговор нема да биде вратен. Друго нешто што вреди да се обрати внимание е важноста на параметарот за доверба. Доколку изработуваме на пример чет бот за поддршка на корисници, треба да се постави минимум праг на доверба на одговор пред одговорот да му се врати на корисникот. Во конкретново сценарио подобро е одговорот да биде: „одговор не е најден“ отколку „странска принцеза“.

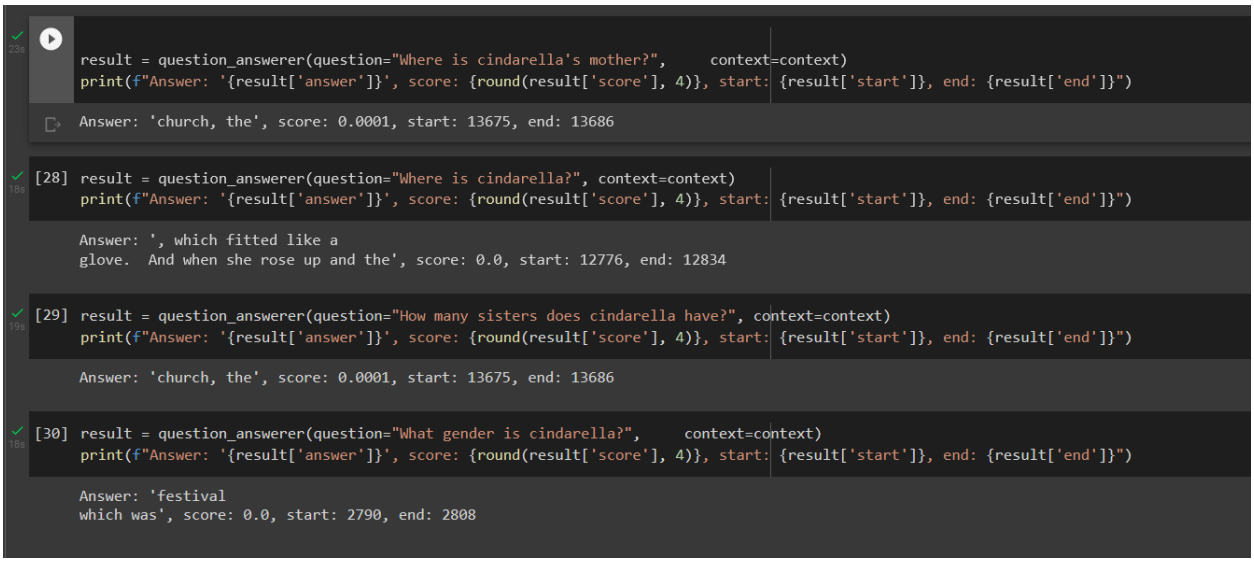
За споредба, може да го направиме истиот експеримент со друг модел. Ќе го земеме SpanBERT за пример. Моделот се наоѓа во истата библиотека од huggingface само инцијализираме друг модел за иста намена. На слика 76 е дадена командата и исходот од истата.

```
from transformers import pipeline
question_answerer = pipeline("question-answering", model='SpanBERT/spanbert-base-cased')

Downloading config.json: 100% 413/413 [00:00<00:00, 3.14kB/s]
Downloading pytorch_model.bin: 100% 205M/205M [00:10<00:00, 41.7MB/s]
Some weights of BertForQuestionAnswering were not initialized from the model checkpoint at SpanBERT/spanbert-base-cased and are newly initialized: ['
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Downloading vocab.txt: 100% 208k/208k [00:00<00:00, 2.71MB/s]
```

Слика: 76

Како и во претходниот случај, го користиме истиот контекст и ги поставуваме истите прашања. На слика 77 се дадени прашањата и одговорите од истите. Одговорите се далеку полоши од оние на DistillBERT. Овој модел е помалку адекватен за намената за која сакаме да го искористиме доколку го користиме без дополнително тренирање.



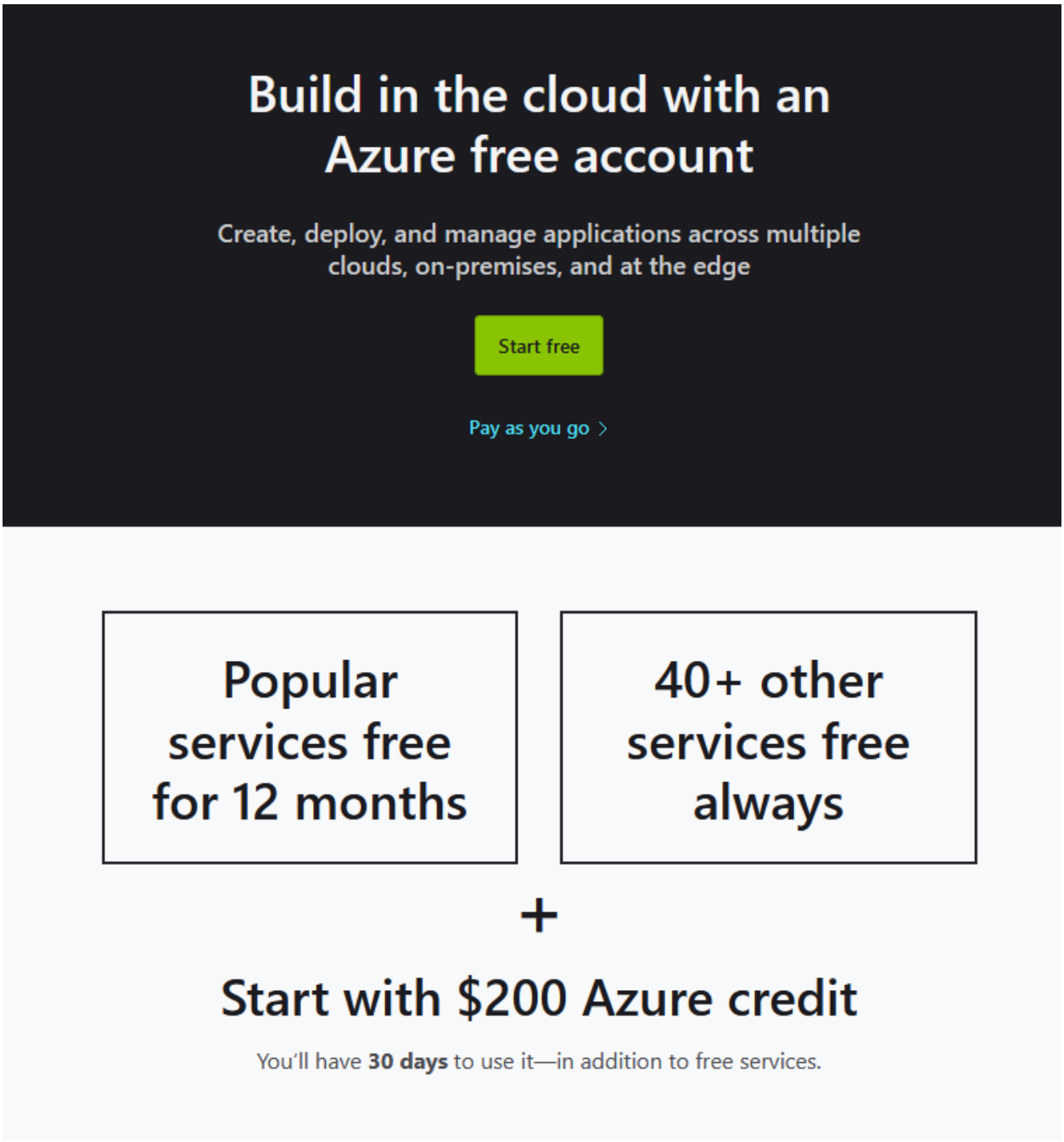
Слика: 77

Во принцип моделите се отворени за дополнително тренирање. Тоа е и неопходно за моделот да биде адекватен за решавање на бизнис проблем. Имено, најчесто даден бизнис домен има свој жаргон. Бидејќи овие модели се тренирани за општа намена, тренирани се на податоци од википедиа и книги, мали се шансите дека имаат разбирање од семантиката на текстот кој може да се најде во документи за менаџирање на пратки на пристаниште или документација од фабрика за преносни инкубатори. Тоа се премногу специфични домени, со жаргон и вокабулар кој можеби не би се сретнал надвор од доменот или пак има комплетно друго семантичко значење. Како што и беше спомнато низ текстот, тренирањето на еден модел бара голема пресметковна моќ, неспоредлива со онаа на просечен персонален компјутер. Другата опција била изнајмување на пресметковна моќ на виртуелни машини на cloud, но тоа е скапа операција. Поради овие причини тренирање на NLP модел нема да биде дел од демонстрациите со практични примери.

За еден модел да може да биде корисен во реалниот свет, истиот треба да може да се интегрира со некаков интерфејс. Не би било воопшто корисно доколку се правиме систем за корисничка поддршка и да очекуваме корисникот да ги пишува прашањата во Python. Најлесен начин за интергација е преикачување на моделот на сервер со REST API. Како што беше напоменато во секцијата за cloud сервиси, постојат десетици ако не и стотици различни начини како тоа да се постигне.

За демонстрација ќе биде даден најминимален начин како тоа да се постигне, а воедно и за најниска цена имајќи го во предвид конкретниот бизнис проблем.

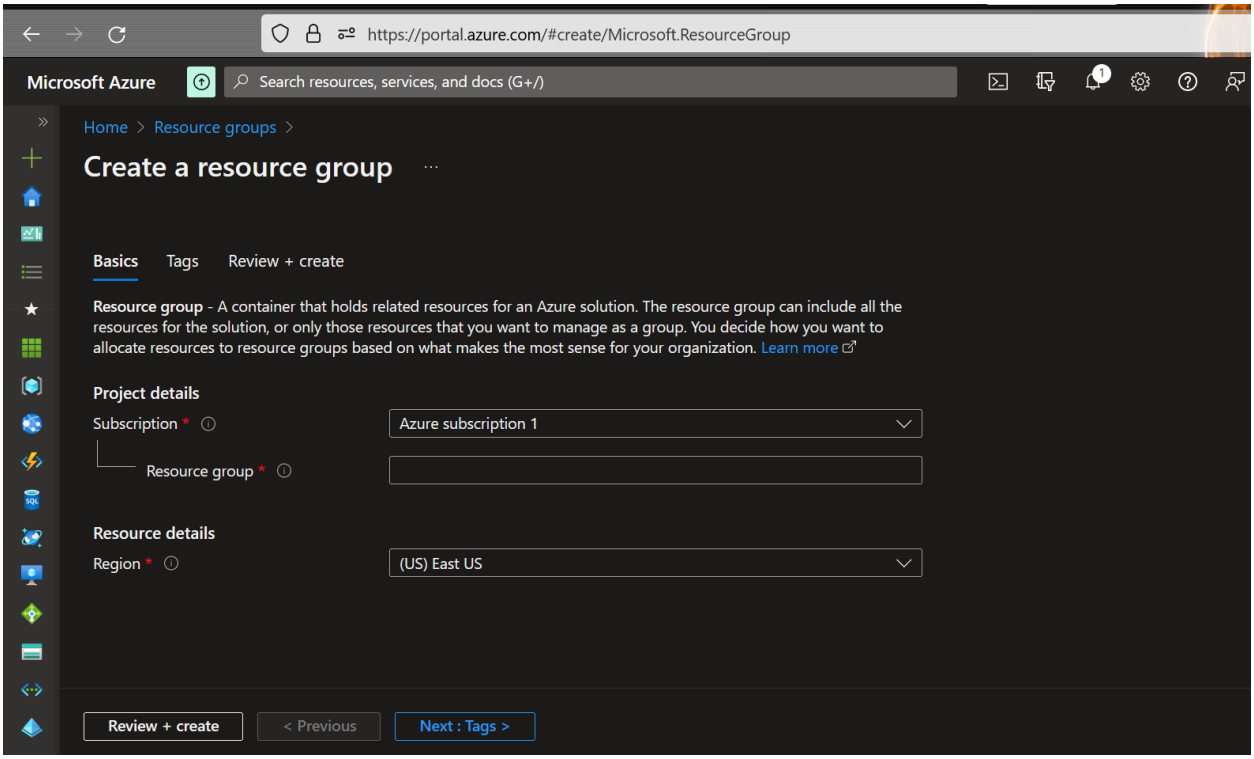
Најпрво треба да се креира профил на Azure. Постои опција за активирање на пробен период. Периодот трае 12 месеци за кое време може да се користат одредени сервиси и дополнително на располагање има кредит од 200 американски долари кои може да се искористат во тек на 30 дена за експериментирање со сервисите кои немаат бесплатен пробен период. По консумацијата на кредитот ресурсите се наплаќаат од поврзаната платежна картичка. На слика 78 е дадена понудата од Azure.



Слика: 78

Битно е да се спомне дека освен времеснското ограничување, има и ограничување на фунционалности.

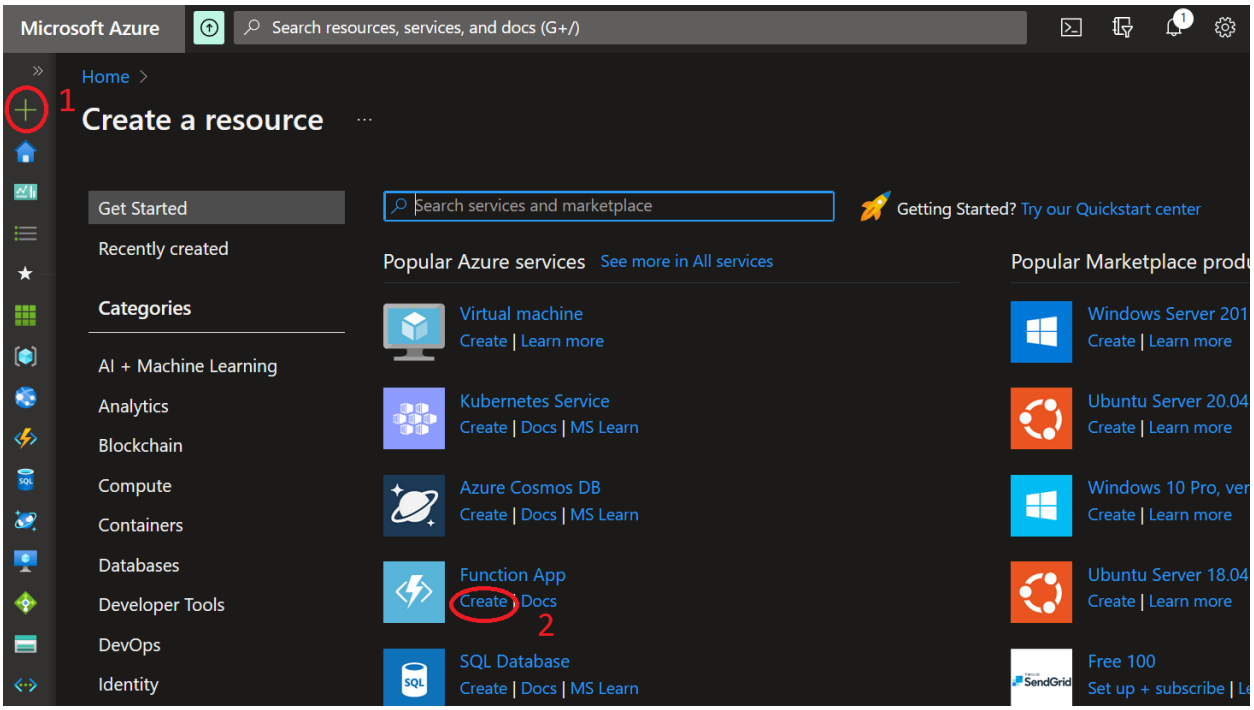
По регистрацијата треба да се креираат ресурси. Постојат неколку начини да се креираат ресурси: преку автоматизирана скрипта, преку командна линија[23] и преку графичкиот интерфејс. На слика 79 е дадена формата за креирање на ресурсна група.



Слика: 79

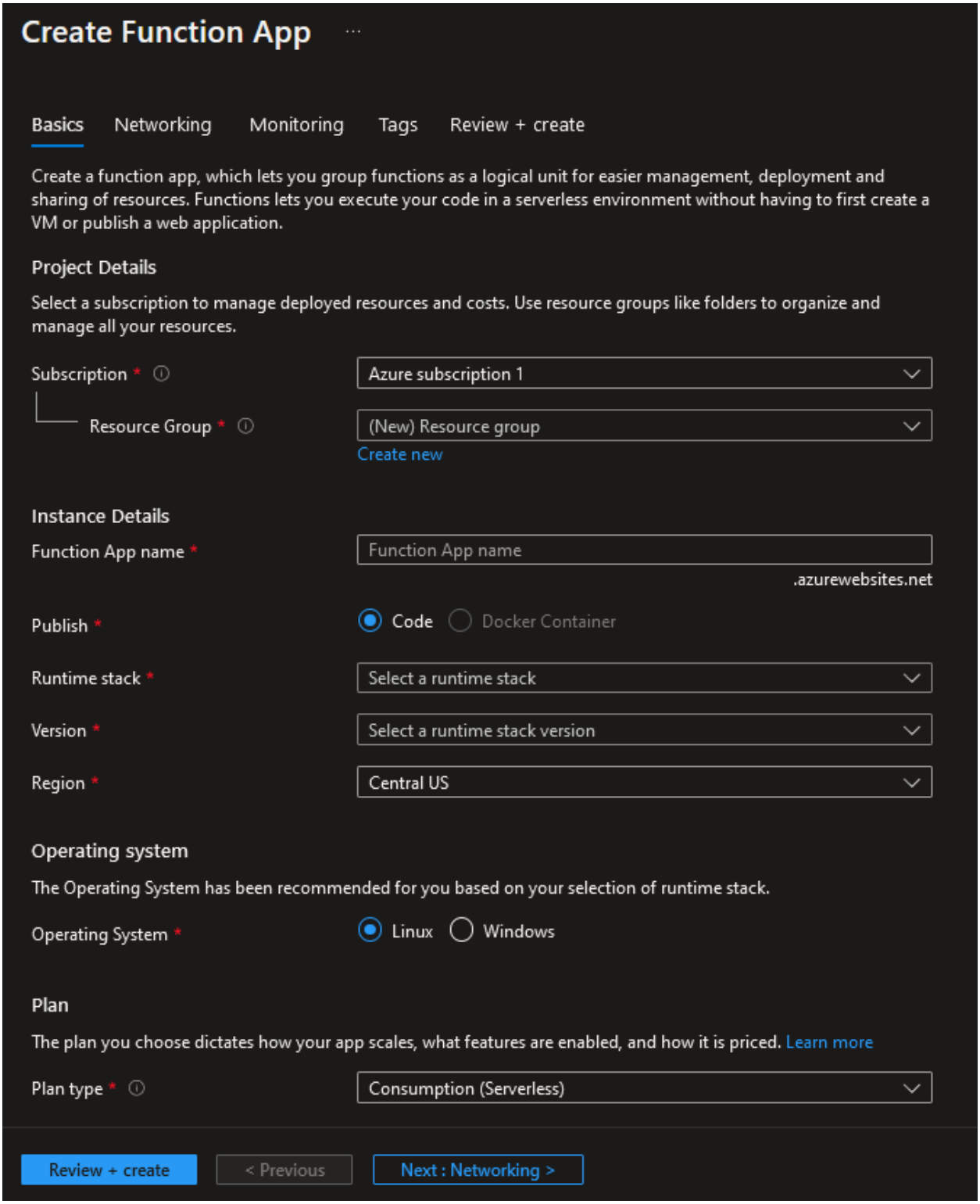
Процесот е доста лесен и јасен, се пополнуваат неколкуте полиња и групата се креира.

Следниот чекор е креирање на Azure Function App. На слика 80 се дадени чекорите.



Слика: 80

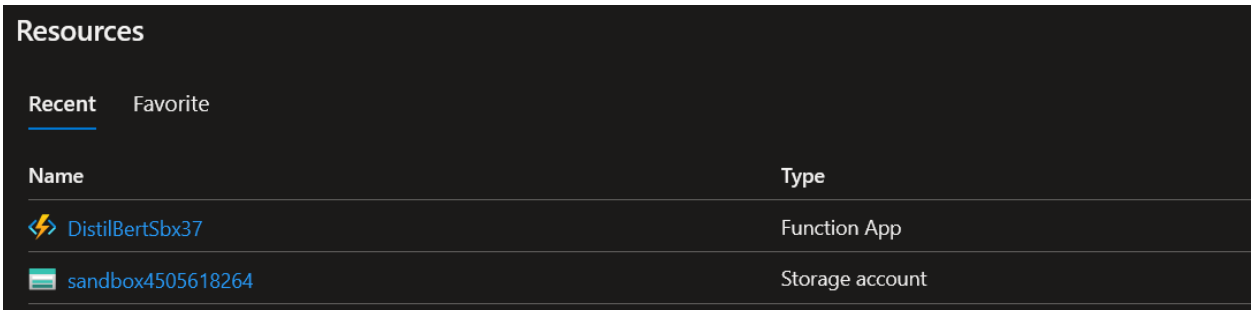
На слика 81 е даден формуларот за креирање на функција.



Слика: 81

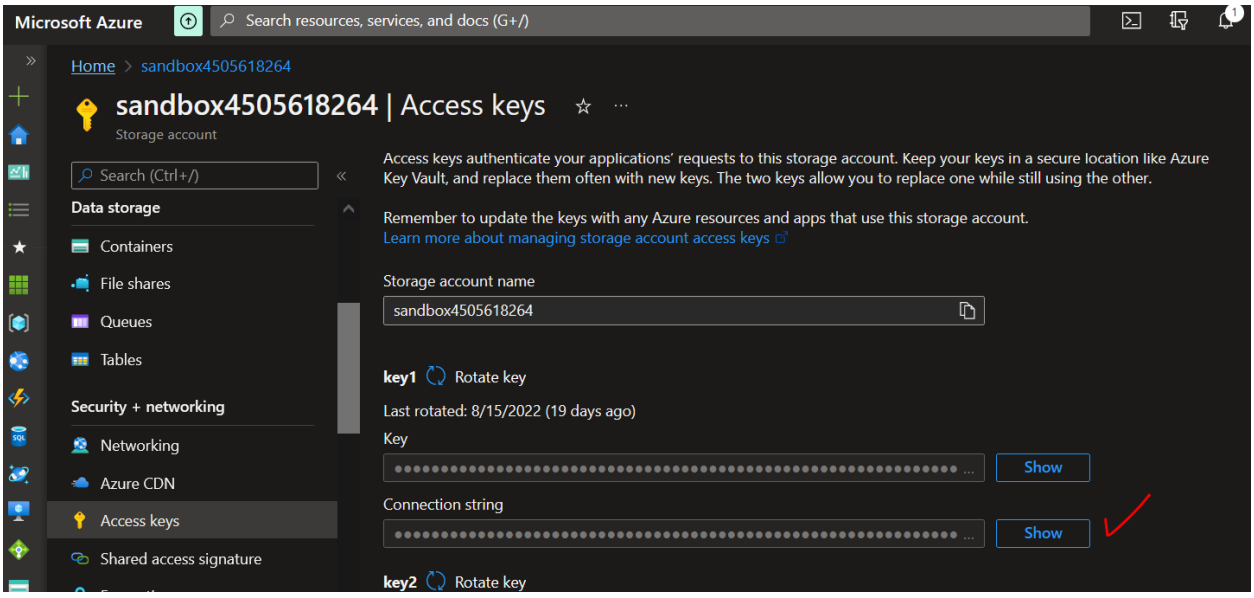
Се одбира соодветната ресурсна група, се доделува име кое мора да биде уникатно во доменот на azure. Како што се гледа на сликата, функционалноста за докер контејнери не е достапна во пробниот период. Бидејќи ќе користиме python во Runtime stack одбираме Python, верзија 3.7 и оперативен систем Linux. Application Insights е одличен систем за мониторирање на ресурси, но во конкретниов случај нема потреба од него и нема го користиме.

Ја креираме функцијата и за неколку секунди ресурсот се појавува во листата. На слика 82 се гледа тоа.



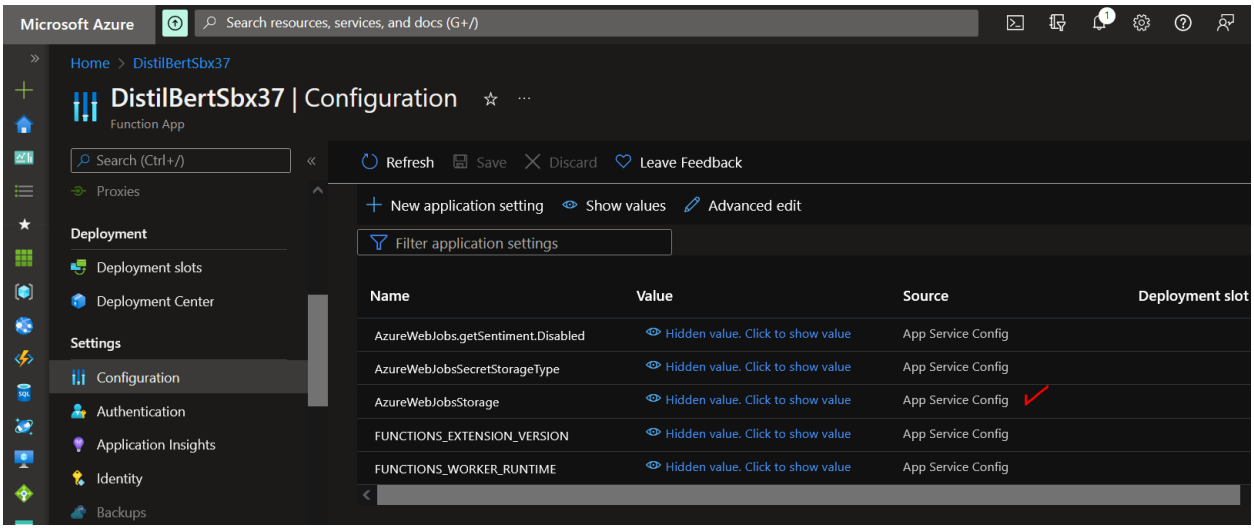
Слика: 82

Функцијата мора да се поврзе со Storage Account за да работи. Акаунтот се креира автоматски и треба само да се поврзе. Тоа исто така може да се направи преку графичкиот интерфејс. Најпрво го копираме клучот од storage account. На слика 83 е дадена формата. Во детаљите за акаунтот -> Security + Networking -> Access Keys -> Connection string. Ја копираме вредноста.



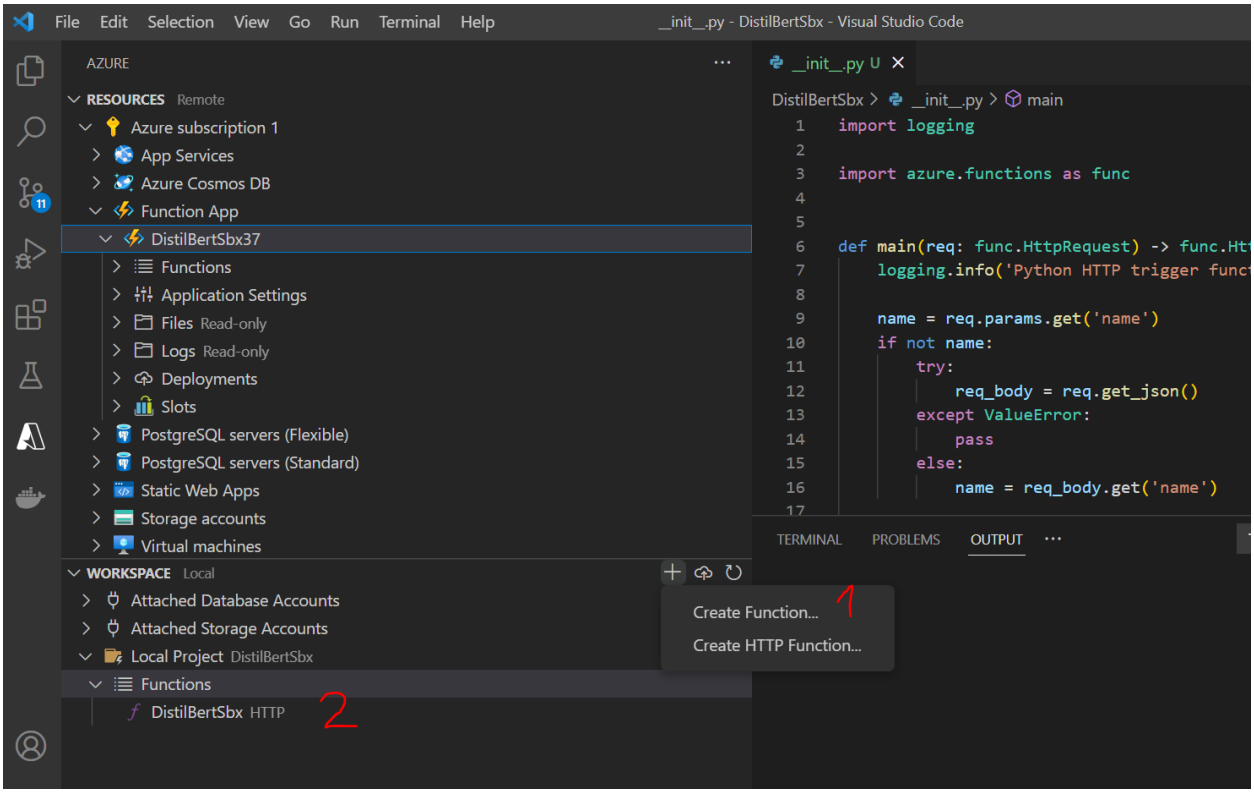
Слика: 83

Потоа во детаљите за функцијата ->Settings -> Configuration се додава клуч со копираната вредност како на слика 84.



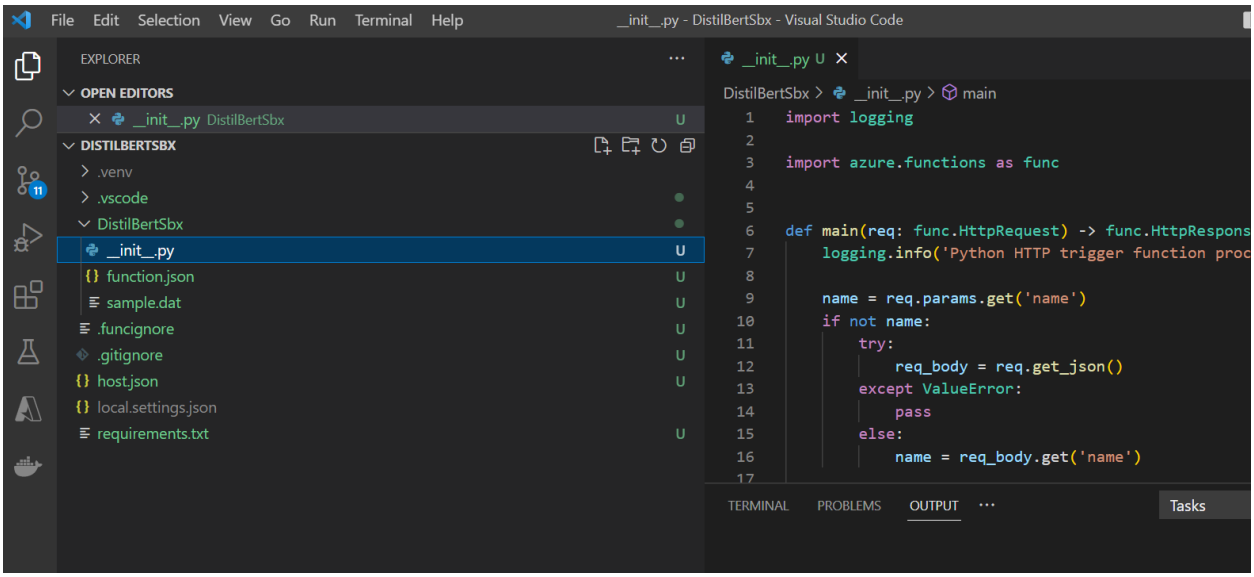
Слика: 84

Следно, треба да ја напишеме функцијата и да го закачиме кодот на креираниот Azure ресурс. Тоа исто така може дасе направи преку графички интерфејс. Може да се користи Visua Studio Code[8] кој е бесплатна алатка за развој на софтвер. Откако се инсталира VS Code, дополнително треба да се инсталираат модулите за развој на функции и менаџмент со Azure ресурси. На слика 85 е дадено како тоа изгледа.



Слика: 85

Со одбирање на Create Function VS Code автоматски генерира темплејтна апликација со сите неопходни датотеки. На слика 86 е дадена структурата на една функција.



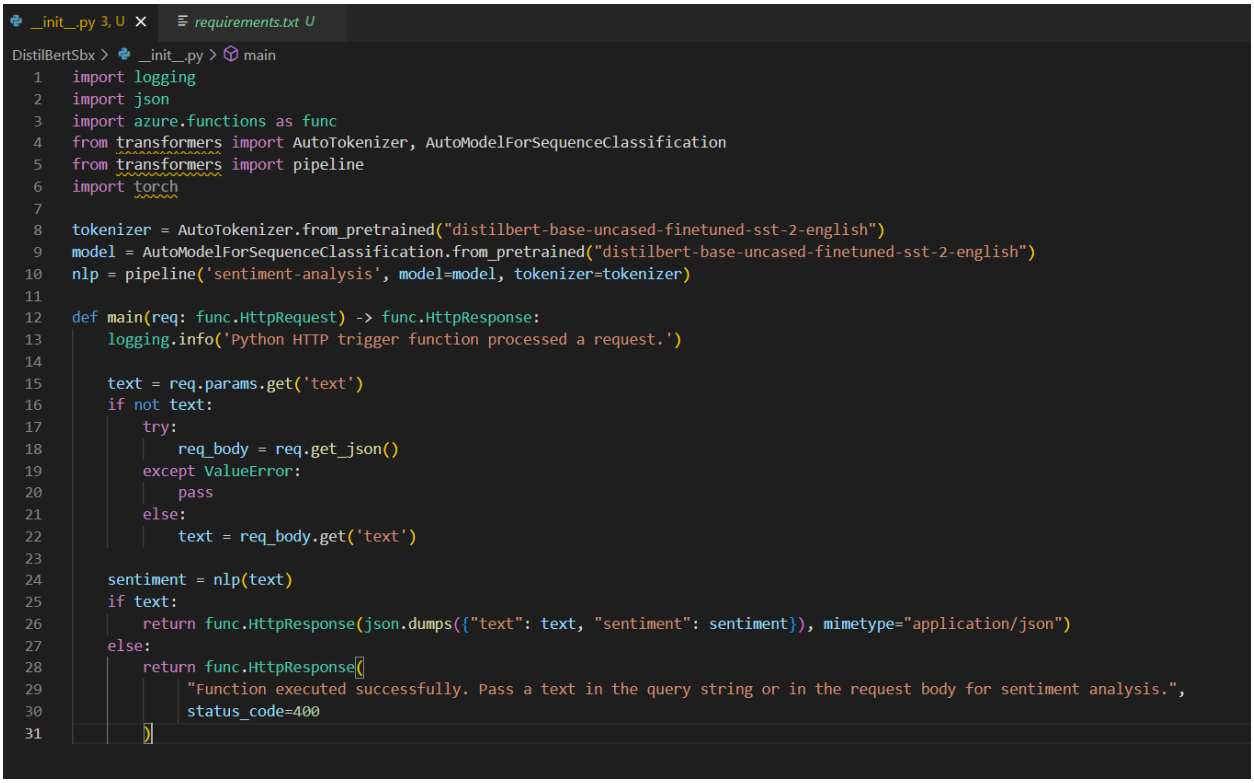
Слика: 86

Кодот кој функцијата го извршува се наоѓа во __init__.py документот. Requirements.txt ги содржи сите библиотеки кои треба да се инсталираат за функцијата да работи, односно тоа се сите надворешни бибилиотеки кои функцијата ги користи. Во function.json се дадени спецификите за самата функција, како на пример тригерот за вклучување. Всушност Azure функциите поддржуваат повеќе типови на тригери:

- Поврзувана со MQ (редица на пораки) функцијата се вклучува секогаш кога пристигнува нова порака.
- Вграден тајмер, така што функцијата се вклучува на специфициран интервал или време од денот
- HTTP тригер – онаа што ние ја користиме – се вклучува при HTTP request односно го опслужува барањето

Host.json документот содржи поставувања кои се аплицираат на ниво на function app. Во принцип една function app може да содржи повеќе индивидуални функции. Local.settings.json документот содржи поставки за локално извршување на функцијата и не се закачува на сервер со останатите документи.

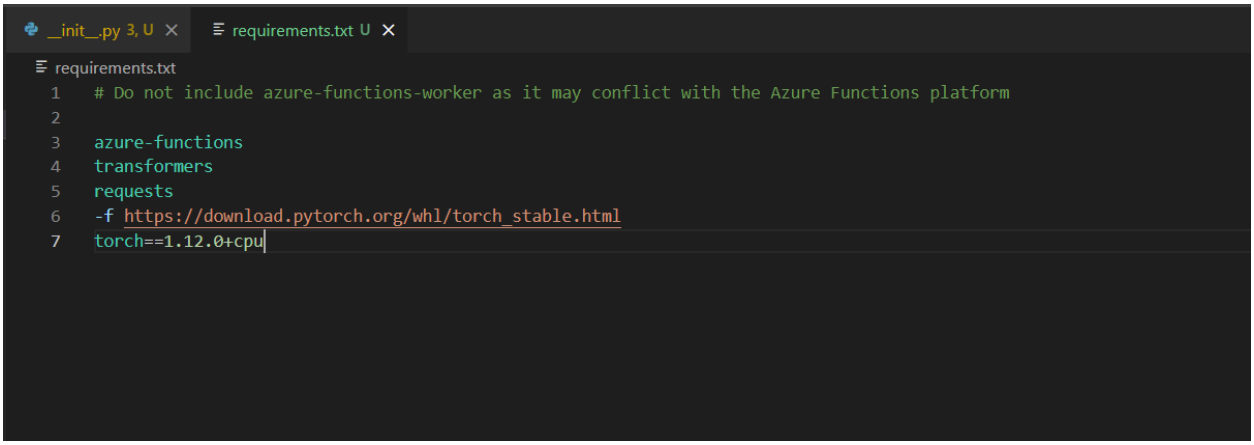
Ја модификуваме python скриптата со кодот даден на слика 87.



Слика: 87

Најпрво импортираме надворешни бибилиотеки. Потоа на линија 8 иницијализираме објект за токенизација. На линија 9 иницијаилзираме пред-трениран модел. На линија 10 го иницијализираме моделот за одредување на чувства. Од линија 12 е дефиницијата на обработката на HTTP барањето. Кога ќе дојде барање со GET глагол, прво проверуваме дали е пратен текст во query string, доколку не е тогаш го бараме во телото на барањето. Ако е најден текстот, тој се испраќа на моделот за евалуација. Моделот го обработува текстот и одредува какво чувство има во него: позитивно или негативно и со каква доверба. Резултатот го враќа назад на клиентот. Доколку не е пратен никаков текст, враќа грешка со код 400.

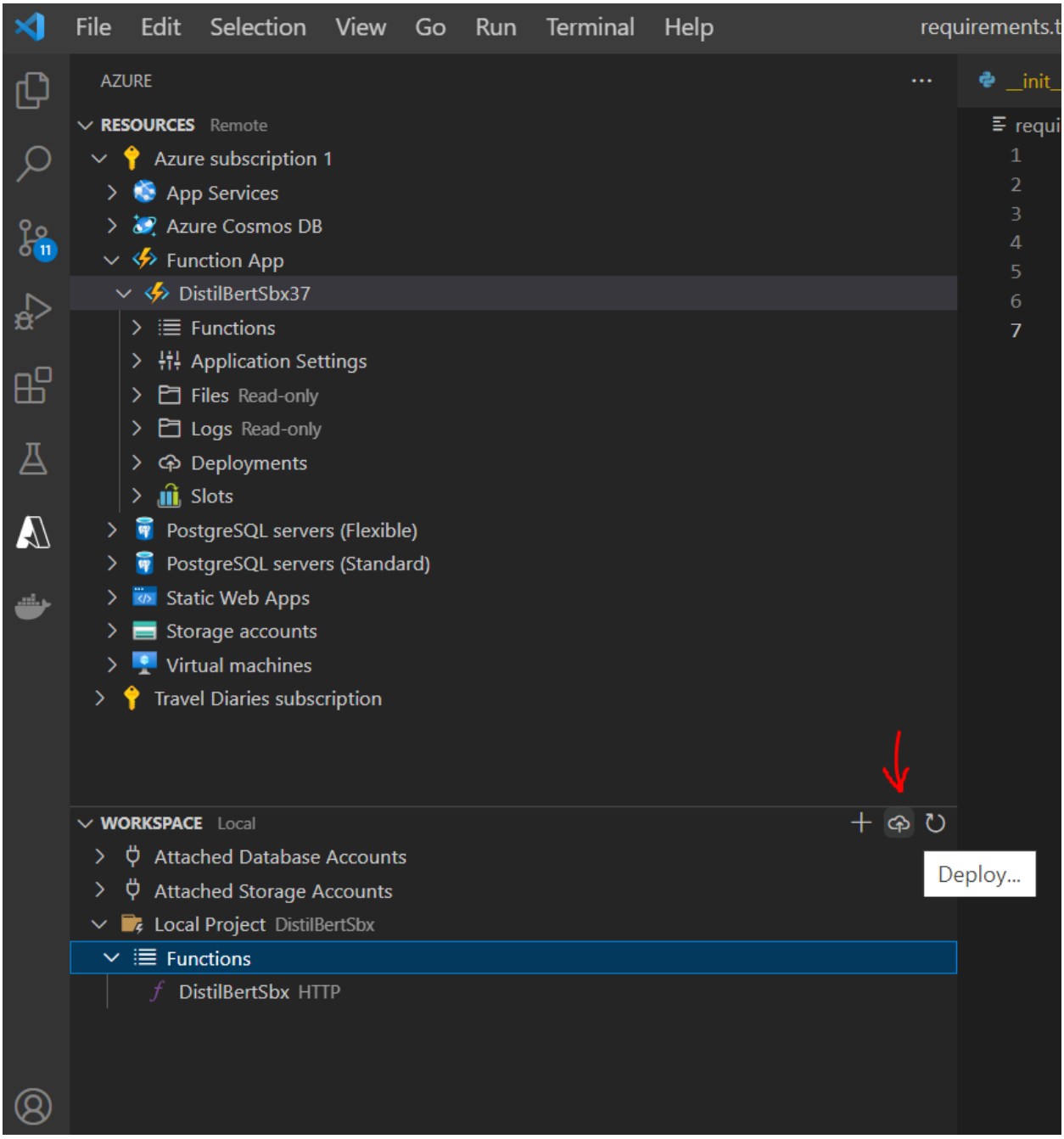
Requirements.txt треба да се измени со кодот даден на слика 88.



Слика: 88

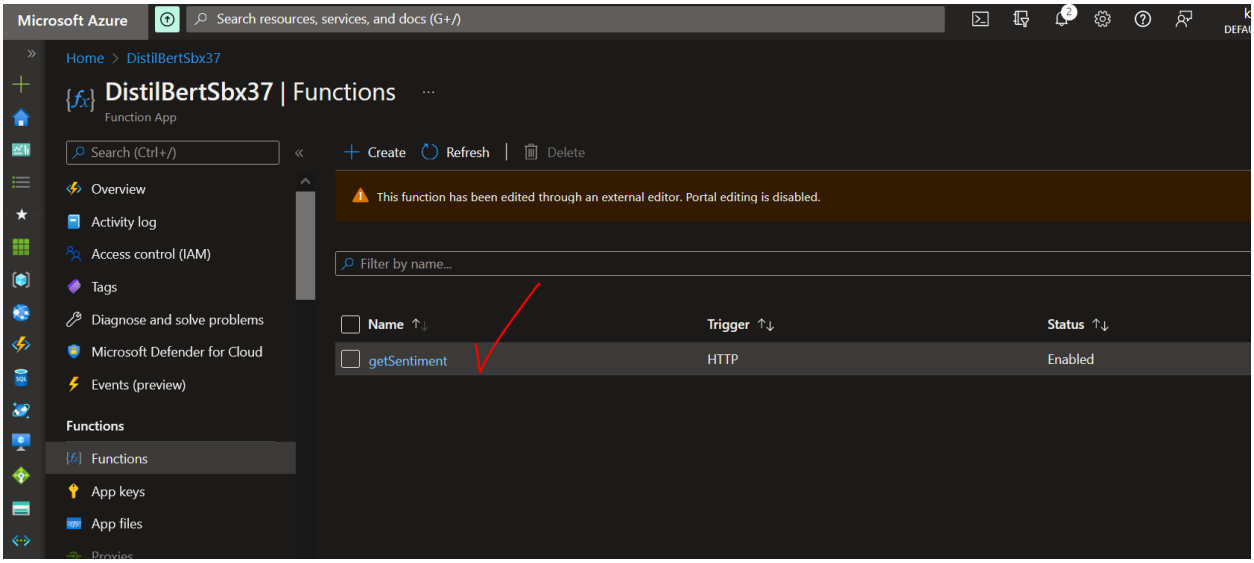
Инсталираме модул што ќе го направи кодот да работи како Azure функција. Потоа ги симнуваме трансформерите од hugging face, потоа инсталираме код за обработка на HTTP барања и на крајот инсталираме pytorch. Pytorch е платформа за машинско учење, за неа мора да специфицираме точна верзија и URL до инсталациските датотеки.

Откако се извршени сите овие чекори, следно е да се закачи кодот на Azure. Преку VS Code UI ова се прави со притискање на копчето Deploy по што се специфицира Azure ресурсот, односно преходно креираниот function app. На слика 89 е даден графичкиот интерфејс.



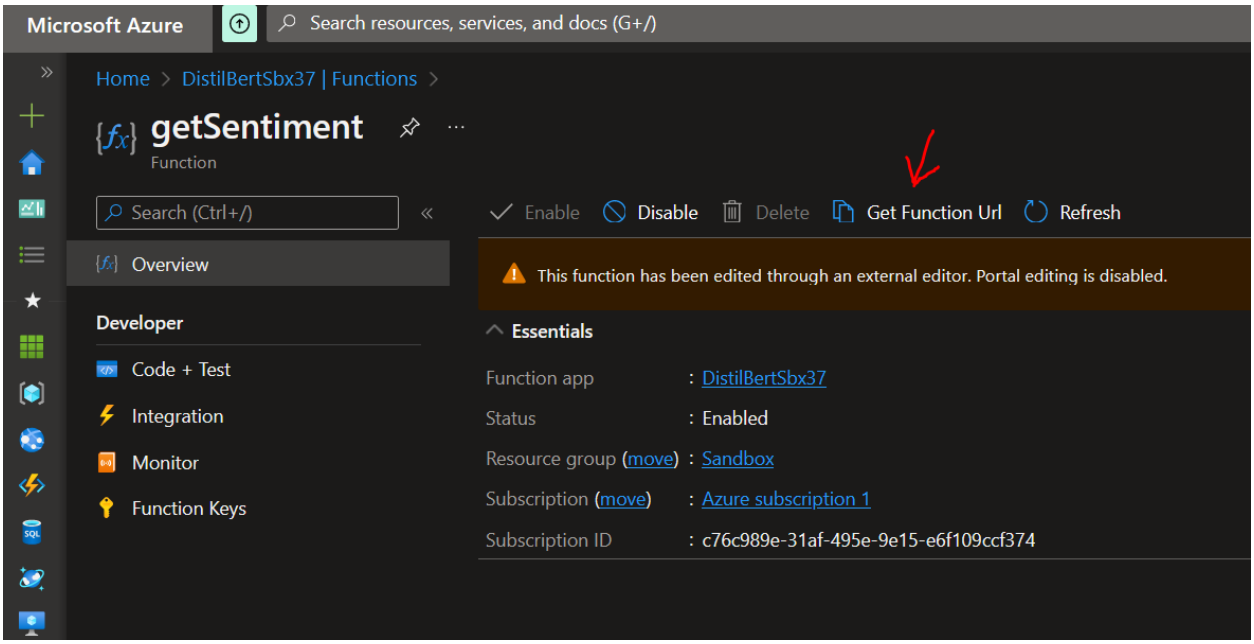
Слика: 89

Откако ќе заврши операцијата, може да се префрлиме во Azure менаџмент порталот во интернет пребарувачот. Ако ги отвориме детаљите за функциската апликација, ќе може да ја видиме функцијата. На слика 90 е дадена формата.



Слика: 90

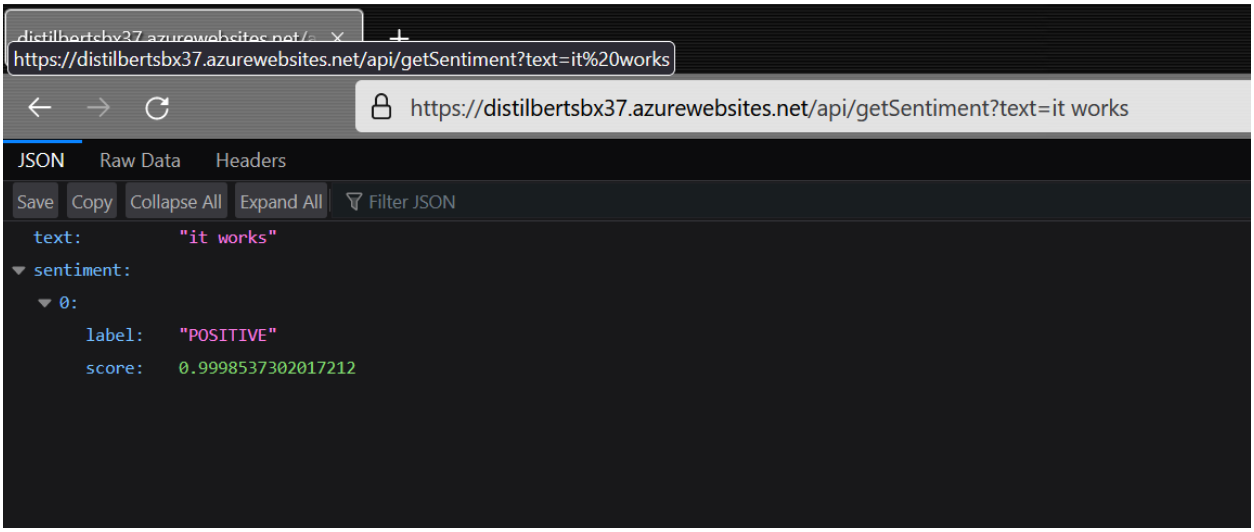
На слика 91 се дадени детаљите од конкретната функција.



Слика: 91

Бидејќи функцијата е закачена како пакет од повеќе датотеки, не е можно да се едитира и тестира директно од овој графички интерфејс. Но ако го овозможиме Application Insights на Monitor секцијата веќе ќе можеме да гледаме логови од извршувањата, кога и колку време траело извршувањето, како било потрошено тоа време и резултатот од извршувањето. Ако притиснеме на копчето Get Function Url ја добиваме адресата на функцијат и можеме да ја тестираме.

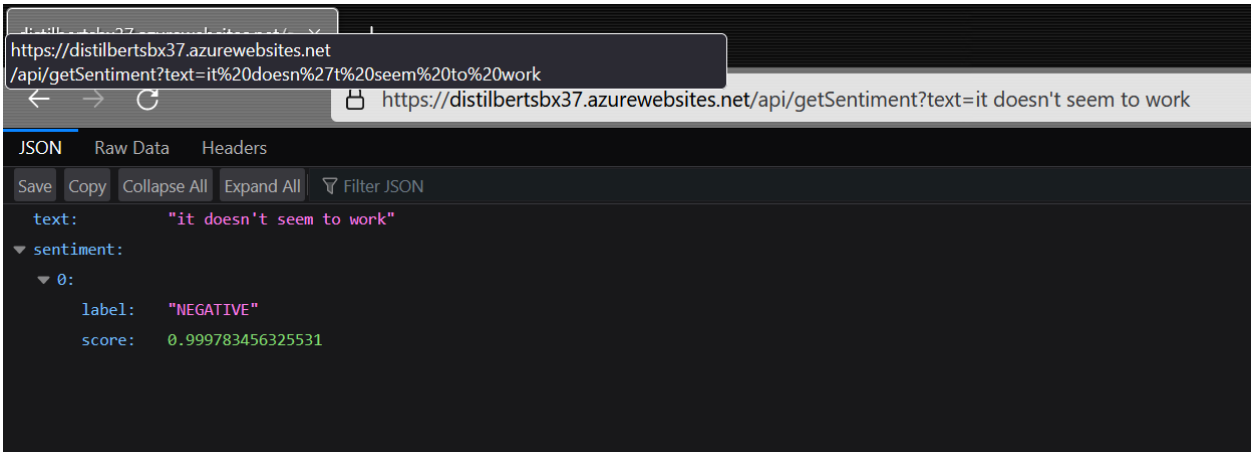
На слика 82 е даден излезот од еден тест.



Слика: 92

„Работи“ е маркирано како позитивно со скоро 1 точност.

На слика 93 е даден излезот од функцијата кога текстот е „Личи дека не работи“. Функцијата ова го маркира како негативно чувство со точност од скоро 1.



Слика: 93

7. Заклучок

Од истражувањето спроведено во доменот на вештачката интелигеницијa, машинското учење и обработката на природни јазици, финалните коментари би биле следните:

- Вештачката интелигенција во доменот на NLP е доста развиена, моделите се со висока ефикасност и нивната практична примена станува нормална, станува подразбирливо решение за реални проблеми. ВИ во NLP е во употреба веќе извесно време, но беше компромисно решение. Сега ситуацијата е поинаква. Моделите се вистински ефикасни и адекватни.
- Машинското учење е за секого. Како заедница е многу отворено за секој кој е заинтересиран да се придружи. Има огромен број на извори и материјали за учење достапни секому. Без разлика на кое ниво е поединецот и за кој аспект од машинското учење конкретно е заинтересиран. За оние кои сакаат да навлеза во детаљите, достапни се оригиналните научни трудови. За оние кој сакаат да експериментираат или брзо да решат некаков проблем, достапни се готови пред-тренирани модели. Нема потреба од губење време за креирање на нешто што некој веќе го направил, сосема возможно е само да се продолжи онде каде што некој запрел.
- ВИ е блиску до секој бизнис, од големи корпорации до мали бизниси. Преку cloud сервисите брзо и лесно вештачката интелигенција може да се инкорпорира во бизнис процесот на било која компанија. Дури и присуството на инженер е опционално. Ова е круцијално за мали бизниси бидејќи не мора да плаќаат скапи човечки ресурси, наместо тоа да се одлучат за решенија со низок код, брза интеграција а сепак задоволителен квалитет.

Вештачката интелигенција е стар концепт кој беше во застој од 60тите до 2000тите години кога длабоките невронски мрежи го вратија во живот. 20 години подоцна, вештачката интелигенција станува се поефикасна и пошироко применлива. Во моментов се чини дека генералната вештачка интелигеницијa се уште не е на повидок. Тоа би била интелигенција без специфична намена, како што е човечкиот мозок. Но сепак, во моментов развојниот тренд е нагорен и нема изгледи дека тоа ќе се смени.

8. Индекс на користени алатки и литература

1. Colab

Cloud сервис за брзо тестирање, експериментирање и развој
Бесплатна и платена верзија
https://research.google.com/colaboratory/faq.html
https://research.google/

2. Jypiter Notebook

Средина за програмирање
Непрофитабилна
https://jupyter.org/

3. Azure Platform

Cloud платформа за SaaS и за развој на софтвер
Платена со тест период
https://azure.microsoft.com/en-us/

4. Pandas

Библиотека за ракување со податоци
Бесплатна
https://pandas.pydata.org/

5. HuggingFace

Платформа за машинско учење
Платена и бесплатна верзија
https://huggingface.co/

6. Python

Програмски јазик за општа намена
https://www.python.org/

7. Pytorch

Платформа за машинско учење
Бесплатно
https://pytorch.org/

8. Visual Studio Code

Средина за развој на софтвер

Бесплатно
https://code.visualstudio.com/

9. Kaggle

Платформа за машнско учење
https://www.kaggle.com/

10. Scikit-learn

Библиотека за машинско учење
https://scikit-learn.org/stable/

11. Seaborn

Библиотека за исцртување графици
https://seaborn.pydata.org/

12. TensorFlow

Платформа за машинско учење
https://www.tensorflow.org/

13. Numpy

Python библиотека за математички операции
https://numpy.org/

14. Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

15. Radford, Alec, et al. "Improving language understanding by generative pre-training." (2018).

16. Liu, Yinhan, et al. "Roberta: A robustly optimized bert pretraining approach." *arXiv preprint arXiv:1907.11692* (2019).

17. Radford, Alec, et al. "Language models are unsupervised multitask learners." *OpenAI blog* 1.8 (2019): 9.

18. Applied Text Analysis with Python by Benjamin Bengfort, Rebecca Bilbro, and Tony Ojeda (2018)

19. Pattern Recognition and Machine Learning by Christopher M. Bishop (2006)

20. Transformers for Natural Language Processing Build innovative deep neural network architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and more by Denis Rothman (2021)

21. AWS, Amazon Web Services - https://aws.amazon.com/machine-learning/

22. IBM cloud services - https://www.ibm.com/cloud/watson-studio

23. Azure CLI - https://docs.microsoft.com/en-us/cli/azure/

24. Lewis, Mike, et al. "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension." *arXiv preprint arXiv:1910.13461* (2019).

25. Lan, Zhenzhong, et al. "Albert: A lite bert for self-supervised learning of language representations." *arXiv preprint arXiv:1909.11942* (2019).

26. Clark, Kevin, et al. "Electra: Pre-training text encoders as discriminators rather than generators." *arXiv preprint arXiv:2003.10555* (2020).

27. Joshi, Mandar, et al. "Spanbert: Improving pre-training by representing and predicting spans." *Transactions of the Association for Computational Linguistics* 8 (2020): 64-77.

28. Sanh, Victor, et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." *arXiv preprint arXiv:1910.01108* (2019).

29. Jiao, Xiaoqi, et al. "Tinybert: Distilling bert for natural language understanding." *arXiv preprint arXiv:1909.10351* (2019).

30. Amazon SageMaker https://docs.aws.amazon.com/sagemaker/latest/dg/whatis.html

31. Pip Installer - https://pip.pypa.io/en/stable/installation/

32. Google AI - https://ai.google/

33. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow by Aurélien Géron (2019)

34. Building Machine Learning Powered Applications by Emmanuel Ameisen (2020)

35. Abraham, Ajith. "Artificial neural networks." Handbook of measuring system design (2005).

36. Ng, Andrew. "Sparse autoencoder." *CS294A Lecture notes* 72.2011 (2011): 1-19.

37. Wolf, Thomas, et al. "Huggingface's transformers: State-of-the-art natural language processing." *arXiv preprint arXiv:1910.03771* (2019).

38. Amazon Rekognition - https://aws.amazon.com/rekognition/

39. Sundermeyer, Martin, Ralf Schlüter, and Hermann Ney. "LSTM neural networks for language modeling." *Thirteenth annual conference of the international speech communication association*. 2012.

40. Agarap, Abien Fred. "Deep learning using rectified linear units (relu)." *arXiv preprint arXiv:1803.08375* (2018).

41. Murtagh, Fionn, and Pedro Contreras. "Methods of hierarchical clustering." *arXiv preprint arXiv:1105.0121* (2011).

42. Gensim - https://radimrehurek.com/gensim/

43. NLTK - https://www.nltk.org/