Projekt na predmet Strojové učenie

Martin Demeter, FMFI UK

2018

V mojom projekte som sa venoval textovej klasifikácií. Na kaggle.com som našiel dataset s názvom 50000 job borad records from Reed UK. Ten obsahuje 50000 riadkov, kde každý riadok predstavuje informáciu o pracovnej ponuke. Konkrétne stĺpce: category, city, company_name, geo, job_board, job_description, job_requirements, job_title, job_type, post_date, 'salary_offered', state. Mojím cieľom v projekte je natrénovať model, ktorý by z popisu pracovnej ponuky - stĺpec job_description, vedel predikovať kategóriu - stĺpec category. Kategória predstavuje oblasť práce pracovnej ponuky.

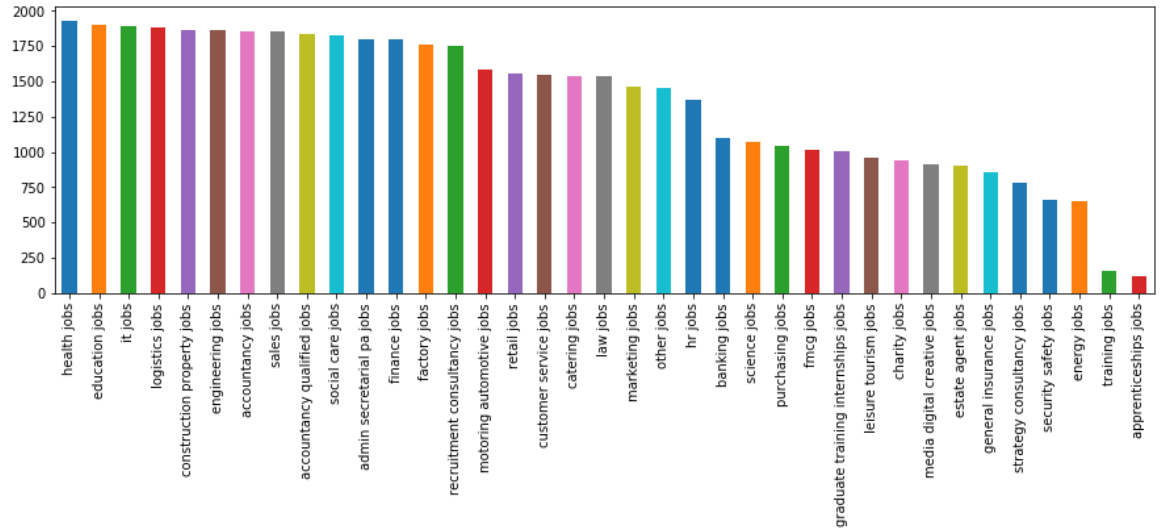
Tento dataset a úlohu som si vybral pretože v práci sa venujeme podobnému problému. Konkrétne naprogramovali sme automatizovaný systém, ktorý dokáže cieliť fb reklamy na pracovné ponuky ľudom, ktorí sa na konkrétnu pracovnú ponuku hodia. Súčasťou systému je detekcia podobnosti pracovných ponúk. Ak užívateľ navštívil pracovnú ponuku na našom job portály, neskôr mu môžeme zobrazovať reklamy na podobné pracovné ponuky. Podobnosť pracovnej ponuky detekujeme už na základe niektorých faktorov, ale textovú anlýzu sme ešte neimplementovali. Preto klasifikácia pracovných ponúk do kategórií(aj s pravdepodobnosťami) nám môže priniesť ďalšie užitočné informácie o podobnosti. Avšak v našom systéme sa nenachádza dostatočne veľká vzroka pracovných ponúk, vďaka ktorým by sme mohli natrénovať úspešný klasifikátor, preto som sa rozhodol využiť tento dataset, ktorý v budúcnosti môžem doplniť o ponuky z iných zdrojov.

Analýza dát

Skôr ako som sa pustil do predspracovania textu, pozrel som sa na to, ako vyzerajú dáta.

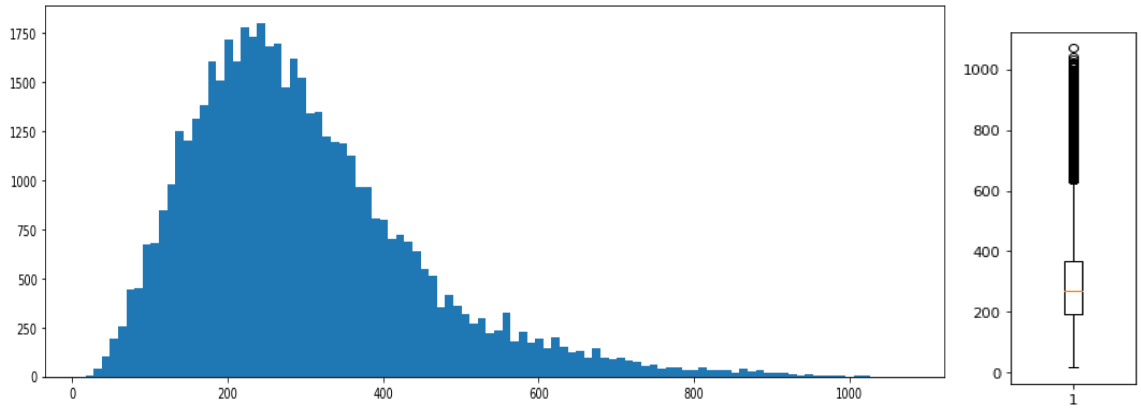
Každá pracovná ponuka patrí do jednej z 37 možných kategórií. Žiaľ, rozdelenie do týchto kategórií nie je rovnomerné, čo je bežným problémom pri práci s reálnymi dátami. Takéto dáta sa nazývajú tzv. Skewed data. Zvlášť pri klasifikácií je dôležité myslieť na to, že viaceré modely môžu mať s takýmto skosenými dátami problém. Dôležité je si tiež uvedomiť, ako správne vyhodnocovať úspešnosť modelov. Napr. ak robíme klasifikáciu toho, či je email spam alebo nie a náš dataset obsahuje len 10% spamov, tak model, ktorý klasifikuje všetky emaily ako nie-spam, dosahuje až 90% presnosť, čo je samozrejme celkom dosť, ale takýto model by asi nebol veľmi užitočný. Preto sa pri práci s takýmto datasetmi používajú iné metriky ako napr. F1 skóre. Tiež veľmi užitočnou pomôckou je vizualizácia tzv. Confusion matrix, ktorá nám dáva predstavu o tom, kde náš model robí chybu.

Neskôr sa ukáže, že modely, ktoré som skúšal v mojom projekte nemali výrazný problém so skosenými dátami. Histrogram nižšie ukazuje rozdelenie pracovných ponúk do kategórií.



Ďalej som sa pozrel na to, ako vyzerajú popisy práce, koľko a aké slová sa nachádzajú v každom popise.

Histogram a box graf nižšie vizualizuje počet slov v popisoch práce. Z box grafu možno vyčítať , že priemerný počet slov je cca 300, maximálny počet slov je cca 1100, 90% popisov má do 600 slov, pričom až polovica popisov práce má cca 180-380 slov. Histogram ukazuje distribúciu počtou slov, kde os x predstavuje počet slov a os y počet pracovných ponúk.



Všetky popise práce obsahuju celkovo 14391380 slov, z toho 375172 je unikátnych. Najpoužívanejšie slová si môžeme vizualizovať tzv. Wordcloudom.

Tu si možno všimnúť, že slovné spojenie “Apply now “ je podozrivo veľké. Pri vypísaní niekoľkých popisov práce som zistil, že veľmi veľa pracovných ponúk začína vetou “Apply now...”. Vo fáze preprocesingu by bolo vhodné toto spojenie vymazať.

Ešte pred začatím preprocesingu som rozdelil dataset na trénovaciu a testovaciu vzroku v pomere 10:8. Testovaciu vzorku som uložil do csv súboru a bude použitá až na úplnom konci projektu. Počas trénovania budem pracovať len s trénovacou vzrokou, ktorú podľa potreby rozdeľujem ešte na validačnú vzorku.

Text preprocesing

Keďže počítač vie pracovať len s číslami, nastáva problém ako reprezentovať text ako čísla. Existuje viacero spôsobov ako mapovať slová do čísel, ale na to aby sme to robili efektívne, je dôležité text predspracovať - normalizovať.

Základnejšie normalizačné techniky sú: konvertovať text len do lowercase, odstrániť interpunkciu, diakritiku a white-spaces, konvertovať čísla v texte na slová alebo ich odstrániť. Ďalej sa zvyknú odstraňovať nepotrebné slova tzv. stop words - sú to slová, ktoré nenesú žiaden dôležitý význam ako zámená, častice, spojky, predložky, citoslovcia. Niekedy sa zvyknú odstraňovať slová, ktoré sú pre náš problém nepotrebné, napr. Ak sa pracuje na zisťovaní sentimentu textu, zachovávajú sa len slová, ktoré nesú nejaký sentiment - predovšetkým prídavné mená. V mojom probléme odstraňujem len tzv .stop words.

Ďalšia užitočná techinka je konverovanie slov na ich základnú podobu. Vďaka tomu vieme zredukovať počet slov, s ktorými model musí pracovať. Existujú dva prísupy. Stemming – prevedie slovo na jeho koreň. Všetky deriváty jedného slova ako sú množné číslo(-s), minulý čas(-ed), priebehový čas(-ing) konvertuje na jeden základný tvar. Podobnou ale zložitejšou technikou je lemmatization – pri lemmatizácií sa berie do úvahy aj význam slova. Každé slovo konvertuje na jeho základný tvar – lemu. Napr. Slovo ‘’better’’ konvertuje dokáže konfertovať na ‘’good’’. Porovnal som oba prístupy a lematizácia sa ukázala ako mierne lepšia.

Po normalizácií textu je potrebné každý záznam vhodne konvertovať do číselnej podoby - konkrétne do vektorovej reprezentácií. Štandardným spôsobom je použiť techniku Bag of words. Najprv sa definuje fixná dĺžka vektora, kde každý prvok reprezentuje jedno slovo z

celého textu. Veľkosť vektora sa rovná počtu unikátnych slov v texte. Vektor pre každý záznam vytvoríme tak, že do vektora zapisujeme koľko krát sa aké slovo vyskytlo v našom zázname. Na to slúži v sklearn trieda Count vectorizer. Ďalším technikou je tzv. Tf-ids, ktorá má výhodu v tom, že slová, ktoré sa nachádzajú vo všetkých záznamoch majú menšiu váhu ako unikátnejšie slová. Vďaka tomu sa pre slová ako ‘’Apply’’ ,’’now’’, ktoré sa pri analyzovaní dát našli v každom popise práce zníži váha. Ukázalo sa, že tf-ids vectorizer funguje lepšie ako count vectorizer. Problémom oboch techník je, že nezachytáva sa význam slov v texte.

Po konvertovaní textových záznamom na vektory som dostal vektory dĺžky 79843. Bolo by ešte vhodné redukovať tento počet atribútov(features), teda odstrániť také dimenzie z vektora, ktoré charakterizujú nepotrebné slová. Na to existuje v sklearn trieda SelectKBest, ktorá na základe skórovacej funkcie vyberie k-najlepších tokenov. Vyskúšal som niekoľko skórovacích funkcií a najlepšie sa osvedčila chi2 funkcia, ktorá na nájdenie najlepších atribútov používa chi-square test. Ukázalo sa, že z celkového počtu 79843 slov, som vybral len 40000 bez toho aby to ovplyvnilo chybu na validačnej vzorke.

Hľadanie modelov a trénovanie.

Keďže pracujem s pomerne veľkým počtom dát(32000 záznamov x 40000 tokenizovaných slov), dal som prednosť vytvorením validačnej vzorky pred cross-validáciou, pretože by to hľadanie správneho modelu so správnymi hyperparametrami trvalo príliš predĺžilo.

Dummy klasifikátor

Pri overovaní úspešnosti modelov je vhodné si určiť spodnú hranicu presnosti modelov pomocou triviálneho modelu. V prípade regresie, najtriviálnejší model, ktorý predikuje hodnotu je výpočet priemeru hodnôt z testovacej vzorky. V prípade regresie to môže byť model, ktorý klasifikuje náhodné triedy. Lepší model by bol, ak by zároveň rešpektoval distribúcie tried. Teda ak ma dataset 90% jednej tiedy a 10% druhej, binárny dummy kvalifikátor by s 90% pravdepodobnosťou určoval ako výslednú tú triedu prvú a s 10% tú druhú. Sklearn má implementovaný takýto klasifikátor pomocou triedy DummyClassifier(strategy='stratified'). Ten na testovacej aj validačnej vzorke dosiahol f1 skóre 0.03. Keďže robíme klasifikáciu do 37 rôznych tried s rôznymi distribúciami je veľmi malá pravdepodobnosť, že sa náhodne pridelí správny počet tried.

Random forrest

Ako prvý skutočný model som vyskúšal RandomForrest. Bez nastavovania hyperparametrov som dostal f1 skóre 0.701. Kedže robíme multi-class klasifikáciu s 37 triedami a máme len 32000 záznamov, na prvý pokus si myslím, že je to celkom prekvapivý výsledok. Na trénovacej vzorke som dostal skóre 0.99. Pomocou grid searchu som skúšal hľadať hyperparametre, ktoré by znížili overfit a zároveň zvýšili skóre na validačnej vzorke. Najlepšie f1 skóre 0.77 som dosiahol s hyperparametrami: počet stromov 150, a maximálna

hĺbka 35. Skúšal som aj iné hyperparametre, ale ukázalo sa, že neviem znížiť overfit bez toho aby sa znížilo skóre na validačnej vzorke. Vyskúšal som aj ExtraTree klasifikátor, ktorý by mal mať väčšiu kontrolu pred overfitingom, ale výsledok bol veľmi podobný. Preto som vyskúšal jednoduchšie modely ako stromy.

SVM

Ďalším pokusom bolo SVC. Použitie rbf kernelu je síce veľmi užitočné , ale keďže ide o pomerne veľký počet dát, trénovanie a hľadanie hyperparametrov by trvalo príliš dlho. Preto som používal iba lineárny SVM. Bez nastavovania hyperparametov, LinearSVC prišlo s chybou 0.79 na validačnej vzorke a 0.98 na trénovacej. Overfitting som skúšal obmedziť nastavením l2 penalizácie, čo znížilo overfit na 0.94 pričom zachoval skóre na validačnej vzorke. Pomocou grid searchu som našiel najlepšie skóre 0.80(0.92) s týmito hyperparametrami: C=1.1, penality=l2, dual=True – rieši duálny problém namiesto primárneho, vhodné ak počet atribútov je väčší ako počet záznamov, loss=hinge, class_weight=balanced = vhodné pre dáta, kde sú triedy v rôznom pomere.

Logistická regresia

Najjednoduchší model prekvapivo dosiahol skóre 0.77 na validačnej vzorke a 0.83 na trénovacej vzorke, čo je najmenší rozdiel z pomedzi všetkých modelov. Skóre som dosiahol s hyperparametrami: C=0.8, solver=saga – ide o variant stochastického gradient descentu, class_weight=balanced.

Neurónové siete

Na to aby som mohol použiť neurónovú sieť z frameworku keras, použil som iný prístup ako konvertovať text do vektorovej podoby. Konkrétne som využil Tokenizer z tohto frameworku. Tá vytvorí z textu slovník dĺžky n a pomocou neho neskôr konvertuje ľubovoľný text do vektorovej podoby. Keďže text, ktorý chceme vektorizovať, môže mať ľubovoľný počet slov, tokenizer nám vráti vektory rôznej dĺžky. Na to aby sme to mohli vložiť ako vstup do neuronovej siete, chceme aby mali vektory rovnakú dĺžku. Na to existuje funkcia pad_sequence, ktorá nastaví vektory na rovnakú dĺžku. Otázkou je, akú veľkosť by sme chceli. Počas analyzovania dát, som zistil, že 90% pracovných ponúk, má dĺžku popisu po lemmatizácií a tokenizácií do 500 znakov, preto optimálna dĺžka vektora by mala mať dĺžku 500. Tie, popisy práce, ktoré sú príliš dlhé odsekne. Nie je ich príliš veľa, preto by to nemalo spôsobiť problémy pri trénovaní.

Architektúra neurónovej siete vyzerá takto:

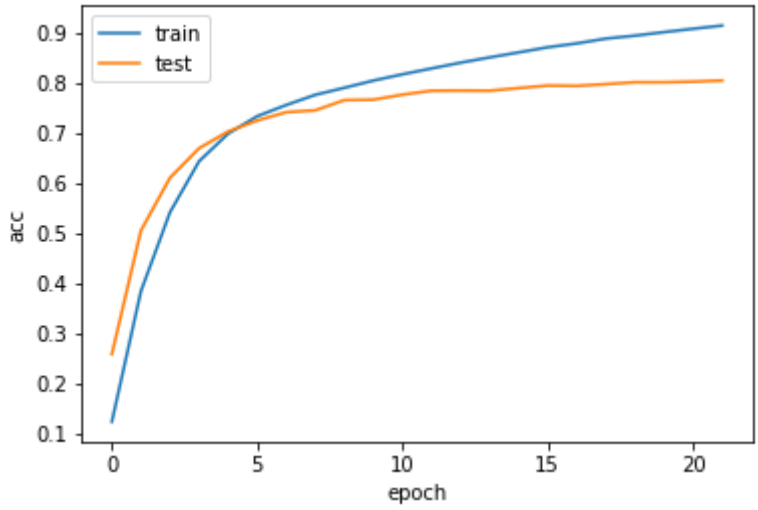
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         (None, 500)               0
_________________________________________________________________
embedding_1 (Embedding)      (None, 500, 64)           2560000
_________________________________________________________________
global_average_pooling1d_1 ( (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 37)                2405
=================================================================
Total params: 2,562,405
Trainable params: 2,562,405
Non-trainable params: 0
_________________________________________________________________
```

Prvá vstupná vrstva obsahuje rovnaký počet neurónov ako veľkosť vektorov.

Druhá vrstva je tzv embedding vrstva. Táto vrstva každé slovo, v našom prípade jeho token, konvertuje na n-dimenzionálny vektor(v mojom prípade je n=64). Tomu sa v NLP hovorí word embedding. Namiesto toho, aby každé slovo bolo reprezentované ako one-hot encoding, čo by znamenalo obrovské plytvanie pamäte(celý vektor obsahuje 0 až na jednu 1), každé slovo sa zakóduje do 64-dimenzionálneho (dense) vektora. Pozícia slova v n-dimenzionálnom priestore závisí od slov, ktoré toto slovo v texte obklopujú. Vďaka tomu je slovo zakódované aj s jeho významom. Slová s podobným významom by sa mali v tomto priestore nachádzať blízko seba.

Ďalšia vrstva predstavuje pooling vrstvu, ktorá redukuje dimenzionalitu, pričom zachováva dôležitú informáciu. Poslednou vrstvou je Dense vrstva veľkosti počtu kategórií(37) a aktivačnou funkciou softmax, čo je štandardom pri klasifikácií.

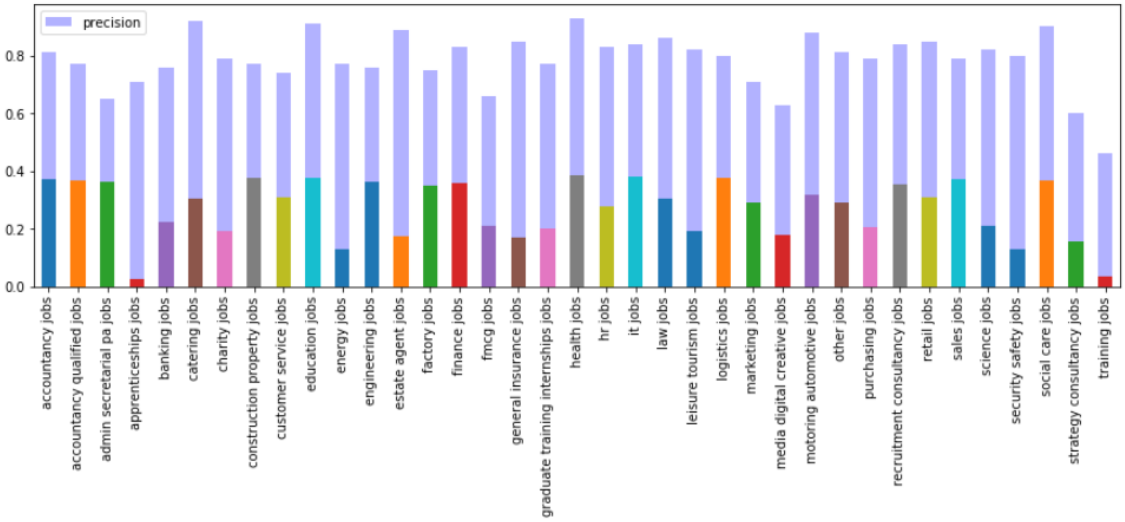
Na trénovanie siete som použil optimizer Adam, implementoval EarlyStopping, ktorý zastaví trénovanie v momente ak sa validačná chyba po niekoľkých epochách nezlepšila, validation split - 0.1 a batch size 512. Dosiahol som f1 skóre na trénovacej vzorke 0.90 a na validačnej 0.79.



Vyhodnotenie modelov

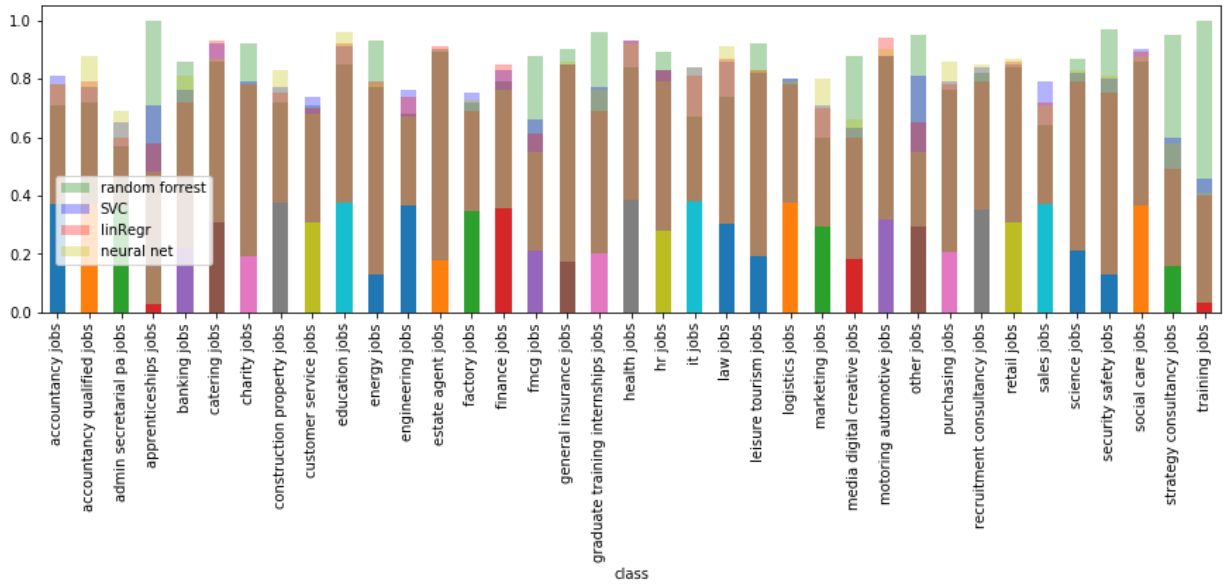
Zdá sa, že sa cez hranicu 0.81 f1 skóre hľadaním lepšieho modelu nedostanem. Preto by bolo na mieste sa lepšie pozrieť na to, kde modely robia chyby.

Prvá vec, ktorá by sa mohla overiť je, či modely robia chyby pri triedach, ktoré sú v menšom zastúpení a či ich modely ''nediskriminujú''. V takom prípade by som musel tieto triedy doplniť o ďalšie dáta. Funkcia classification_report porovná predikované triedy so skutočnými a vráti informácie o úspešnosti pre každú triedu. Výsledok sa dá vizualizovať napr. takto pre SVM model:



Modrou farbou je znázornené aké percento popisov práce bolo úspešne klasifikovaných, zatiaľ čo farebné stĺpce znázorňujú zastúpenie konkrétnej triedy vo vzorke(ide len o vizualizáciu, hodnoty na y-osi sú nerelevantné). Tu možno vidieť, že aj triedy ktoré sú minimálne zastúpené môžu byť lepšie klasifikované ako triedy, ktoré sú silnejšie zastúpene(viď. Apprenticeship jobs vs admin secretarial pa jobs).

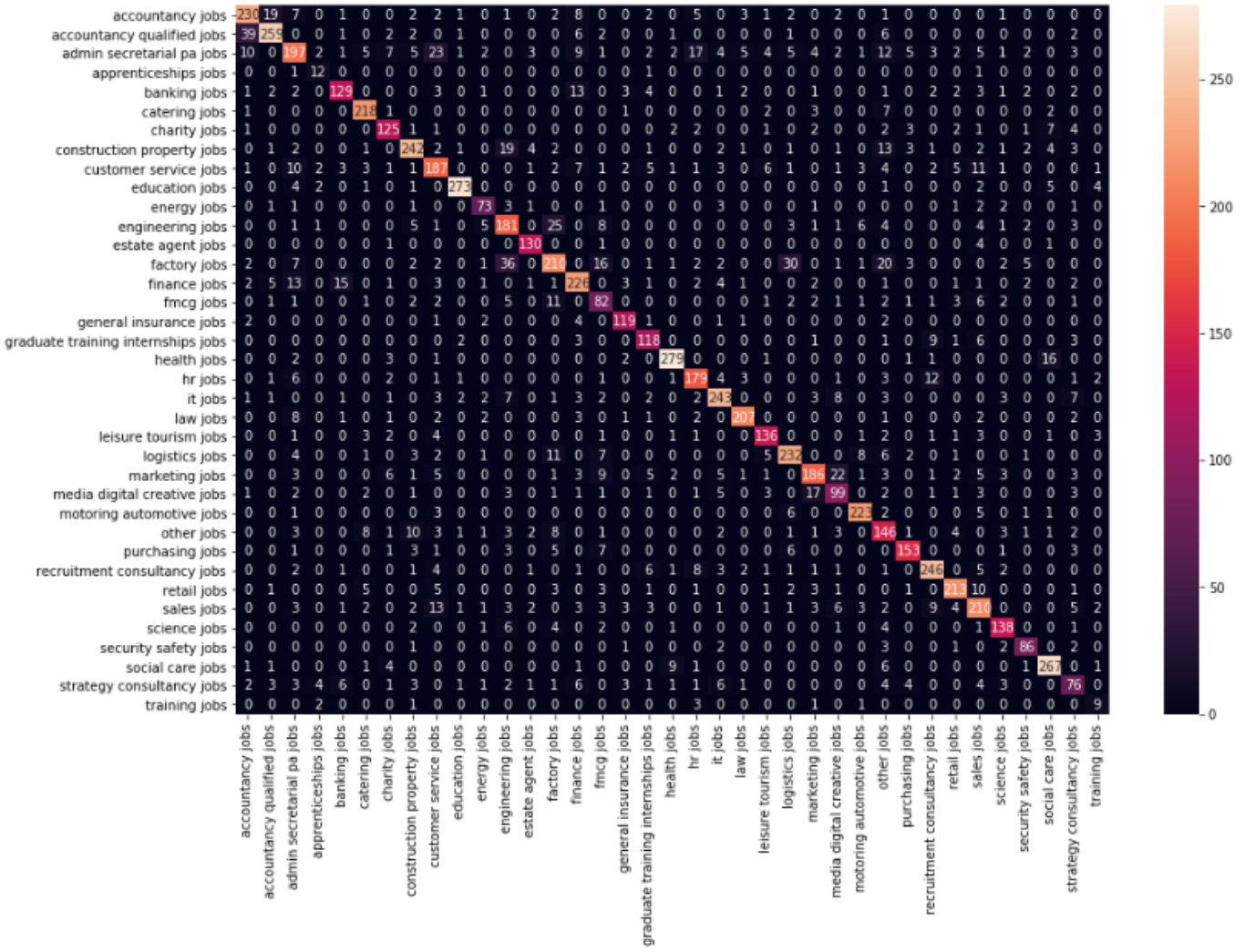
Pre porovnanie modelov zobrazím všetky modely do jedného grafu:



Tu možno vidieť, že random forrest úspešnejšie klasifikoval triedy, ktoré sú minimálne zastúpené. Bližšia analýza ale ukázala, že random forrest klasifikoval len o niekoľko desiatok popisov lepšie a graf skreslil tento výsledok len tým, že zastúpenie týchto tried je malé.

Skúsil skombinovať modely, tak aby spoločne dávali lepší výsledok. Táto technika sa volá Ensemble learning. Na podobnom princípe(Collective wisdom), funguje napr. Random Forrest. Ja som implementoval 2 techniky a to Hard voting a soft voting. Pri hard votingu je výsledná predikcia tá trieda, ktorá bola predikovaná najväčším počtom klasifikátorov. Pri soft votingu je výsledná taká trieda konkrétneho predikátora, ktorý predikoval s najväčšou pravdepodobnosťou. Tu je dôležité aby predikátory dokázali predikovať pravdepodobnosti. SVM tieto pravdepodobnosti defaultne neponúka, preto som tento predikátor nepoužil. Sklearn ponúka implementáciu pomocou triedy VotingClassifier, ale keďže moja neurónová sieť nie je trieda z frameworku sklearn, musel som naprogramovať vlastnú implementáciu. Žiaľ, nedosiahol som lepšie výsledky, hard voting dosiahol f1 skóre 0.80 a soft voting 0.79.

Keďže moja neurónová sieť dosahuje najlepší výsledok, ak zoberiem do úvahy pomer medzi skóre na trenovacej a validačnej vzorke, bližšie som sa venoval len tomuto klasifikátoru. Skvelý nástroj na analyzovanie klasifikátorov je vypísanie confusion matrix:



Y-ová os predstavuje skutočné hodnoty, x-ová predstavuje predikované hodnoty. Z matice možno vyčítať, že model si často zamieňa “accountancy jobs” za “accountancy qualified jobs”, ‘’banking jobs’’ za ‘’finance jobs’’, ‘’customer service jobs’’ za ‘’admin secretarial jobs’’, ‘’customer serivce jobs’' za ‘’sales jobs’’, ‘’engineering jobs’’ za ‘’factory jobs’’, ‘’factory jobs’’ za ‘’logistic jobs’’, ‘’factory jobs’’ za ‘’other jobs’’, ‘’hr jobs’’ za ‘’recruiments consultancy jobs’’, ‘’marketing jobs’’ za ‘’medial digital creativ jobs’’. Už podľa názvu odvetí je zrejmé, že niektoré odvetia spolu zdieľajú veľký počet podobných slov a v niektorých job portálov sú tieto odvetia väčšinou zlúčené. Preto páry odvetí “accountancy jobs”-“accountancy qualified jobs”,‘’hr jobs’’-‘’recruiments consultancy jobs’’, ‘’banking jobs’’-‘’finance jobs’’ by bolo teda rozumné zlúčiť. Vďaka tomu sa podarilo zvýšiť f1 skóre na 0.81.

Testovanie

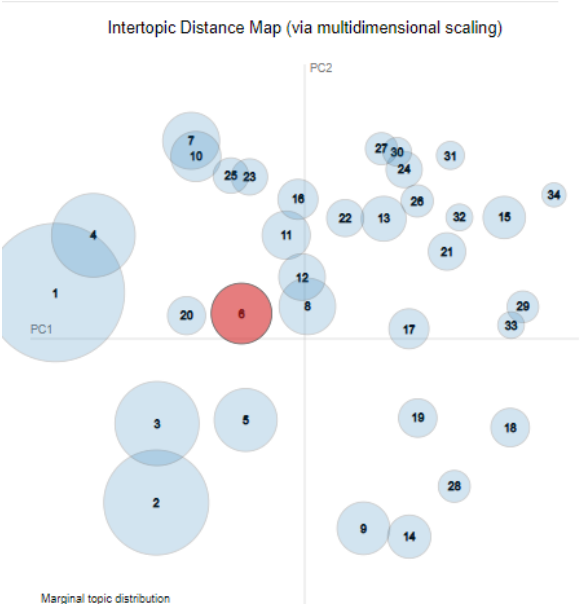
Skúšal aj zložitejšie modely ako LSTM siete a komplikovanejšie neurónové siete, ktoré som našiel na internete, ale keďže som si ich dostatočne nenaštudoval a ani som na nich nedosiahol lepšie skóre, nezakomponoval som ich do projektu. Preto som sa rozhodol uspokojiť sa s jednoduchou neurónovou sieťou, ktorá dosiahla dostatočné skóre a rozdiel medzi trénovaciou a testovaciou vzorkou nie je taký veľký.

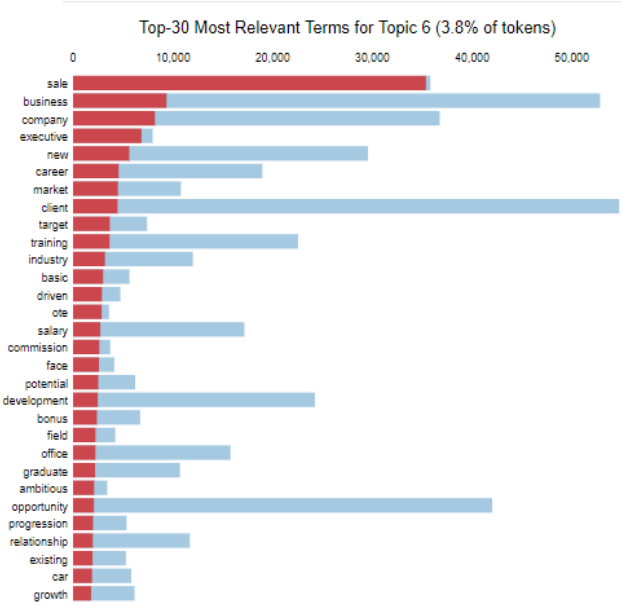
Na testovacej vzorke neurónová sieť dosiahla skóre 0.82, čo je ešte lepšie skóre ako na validčnej vzorke. Často sa stáva, že na testovaciej vzorke sa dosiahne o niečo menšie skóre ako na validačnej, čo je spôsobné prispôsobeniu modelu na validačnú vzorku počas hľadania hyperparametov. Tu takýto problém nenastal a vysvetľujem si to tým, že neurónová sieť bola natrénovaná na celej trénovacej vzorke(bez zmeny hyperparametrov), čím sme zvýšili veľkosť dát o ďalších 8000 záznamov. Posledným pokusom bolo rozšírenie textu o ďalšie dva stĺpce - job_title a job_requirements. Ku každému job_description sa jednoducho prilepil text z týchto dvoch stĺpcov. Ide len o mierne zväčšenie textu. Bez akéhokoľvek preprocesingu model dosiahol skóre 0.8282.

Zdá sa, že nie je problém v modeloch, ale s počtom dát, šumom a lepším preprocesingom. Myslím si, že F1 skóre 0.828 pri klasifikácií do 34 tried na vzorke veľkosti 40000 záznamov len s použitím jednoduchšieho preprocesingu a jednoduchších modelov nie je vôbec zlé. Zároveň sa ukazuje, že akýmkoľvek zvýšením počtu dát sa chyba rýchlo zmenšuje. Preto v budúcnosti sa ako prvé budem pokúšať získať dáta z iných datasetov. Ak to nebude stačiť, skúsim pokročilejšie modely preprocesing a použitie už natrénovaných word embeddingov.

Bonus

Knižnica Gensim ponúka jednoduchý framework na prácu s NLP. Jednou so zaujímavých funkcií, ktoré ponúka je tzv. Topic modeling, ktorý dokáže najsť rôzne témy v texte. Skúsim som text obsahujúci všetky popise práce znovu rozdeliť na 34 tém - oblastí práce. Keďže som mal už predspracovaný text, stačilo málo riadkov kódu. Na výstupe som dostal interaktívnu vizualizáciu, kde sú v kruhoch v ľavej časti nájdené témy a v pravej časti sú slová, ktoré charakterizujú túto tému. V témach sa dajú nájsť aj také, ktoré zo slov môžem identifikovať ako téma finančníctvo , marketing, inžinierstvo, hr a recruitment, obchod ale aj také, ktoré neviem rozdeliť do oblastí práce. Vďaka takejto vizualizácií možno v budúcnosti lepšie pochopiť, ktoré slová sú si tematicky podobné.





Link k interaktívnej vizualizácií:

https://www.kaggleusercontent.com/kf/9517578/eyJhbGciOiJkaXIiLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0..azliGLJoWITG_6dLbC6yAA.FyKgw6C9qLkUi87UUWDMqX2i8zF7yo-nOVC8gVGBohD4bPkJf5hzzfv0euZ_rCDkek6wCEC3ZURxvOO0J7RmWYxjDzDwmothvDgciq_FnfbPr3XKFw6Oiz9rFRf78Gi1OYPd9Hb-sAFubBDT_VUeBg.cwyzCpfZIwELd2oJuh_FPQ/lda.html#topic=6&lambda=1&term=

Zdroje

https://www.kdnuggets.com/2018/03/text-data-preprocessing-walkthrough-python.html

https://www.kdnuggets.com/2017/12/general-approach-preprocessing-text-data.html

https://medium.com/@datamonsters/text-preprocessing-in-python-steps-tools-and-examples-bf025f872908

https://www.kaggle.com/eliotbarr/text-classification-using-neural-networks

https://towardsdatascience.com/how-a-simple-algorithm-classifies-texts-with-moderate-accuracy-79f0cd9eb47

https://towardsdatascience.com/named-entity-recognition-with-nltk-and-spacy-8c4a7d88e7da

https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/