**Porovnanie vybraných algoritmov**

**pre textovú klasifikáciu**

**Abstrakt**

Cieľom je zistiť pri ktorých typoch textových dát a v akých situáciách je vhodné použitie vybraných algoritmov, zistiť ich výhody, nevýhody, porovnať rýchlosť a presnosť. Boli použité algoritmy Naivného Bayesa a Support Vector Machines pri ktorých sa použili dva rôzne datasety s rôznou početnosťou. Výsledky projektu niesú dokončené a porovnáva sa len na základe nameraných presností predikcie testovacích dát.

Autor: Peter Markuš

Akademický rok: 2017/2018

**Použité dáta**

Na porovnanie algoritmov SVM a NB boli použité dva datasety prebraté z internetu ktoré vypovedajú o rozličnej obtiažnosti klasifikácie. Prvý ľahší dataset(20news-18828) obsahuje kategórie z veľmi rôznych oblastí ako je medicína, šport, obchod, počítače a pod. Druhý ťažší dataset (ohsumed-all) obsahuje len oblasť medicíny pozostávajúca z rôznych diagnóz chorôb. Pomer trénovacej a testovacej množiny je 3:1 a je nastaviteľný v implementácii programu. Bližšie informácie o datasetoch a ich spracovaní je nižšie v experimentálnom vyhodnocovaní.

**Kontext**

Naivný Bayesovský klasifikátor je veľmi jednoduchý na implementáciu a je výpočtovo lacný a rýchlejší oproti iným klasifikačným algoritmom. Jednou z veľkých nevýhod je ten, že nefunguje najlepšie pri korelovaných atribútoch. Aj keď je pomerne postačujúci na klasifikáciu textových dát, slová sú si závislé a preto niekedy máva horšie výsledky ako iné diskriminačné algoritmy akými je napr. SVM, ktorý má vyššiu klasifikačnú presnosť. Ukázalo sa, že SVM je jeden z najefektívnejších textových klasifikačných metód v porovnaní s ešte inými rôznymi algoritmami zamerajúce sa na učenie s učiteľom. Pokiaľ je však splnený predpoklad podmienenej závislosti medzi atribútmi, klasifikátor Naivného Bayesa môže mať lepšie výsledky ako SVM. [4, 2]

**Popis a zdôvodnenie použitých metód**

**Support Vector Machines –** jemetóda, ktorá je vhodným kandidátom na klasifikáciu textových dát. Dokumenty spracúva vo vysoko rozmernom priestore a dokáže nepotrebné atribúty eliminovať. Tieto výpočty sú však komplexné a trénovanie často krát trvá dlhšie ako u iných klasifikačných algoritmov a zaplňuje taktiež množstvo počítačovej pamäte. Používa protekciu proti preučeniu, ktorá nemusí nutne závisieť od počtu atribút. Väčšina textových klasifikačných problémov je možné lineárne separovať a myšlienka SVMiek je práve nájsť túto lineárnu separáciu. [4, 1]

Pri používaní iných kernelových funkcií pri veľkom množstve textových atribút sa dá spozorovať, že ich tvar je často lineárny a zbytočne sa ich komplexita používa napríklad aj prostredníctvom hľadaním správneho parametra γ. Nemusí to tak byť vždy, je však vždy vhodné najprv skúsiť lineárny kernel. [5]

V knižnici scikit-learn je možné použiť 2 alternativý lineárneho kernelu. SVC(kernel=‘linear‘) a LinearSVC(), ktorá bola použitá kvôli lepším výsledkom. Ich rozdiely spočívajú v troch zásadných bodoch.

1. Chybová funkcia u LinearSVC minimalizuje kvadratickú chybu. Je však možné definovať túto chybovú funkciu na štandardnú chybu cez loss=’hinge’ tak ako tomu je u obyčajnom SVC s lineárnym kernelom
2. LinearSVC používa metódu multiklasifikačnej redukcie One-vs-All (alebo One-vs-Rest). SVC používa One-vs-One.
3. LinearSVC používa odhady liblinear ktoré penalizujú zachytenie narozdiel od SVC, používajúci odhady libsvm ktoré ich nepenalizujú. Liblinear odhady sú optimalizované pre lineárne prípady a konvergujú oveľa rýchlejšie pri väčšom množstve dát. V konečnom dôsledku je treba poznamenať, že LinearSVC kvôli tomu nieje úplne lineárny kernel. [6]

**Naivný Bayesovský klasifikátor -** naivný sa volá preto, lebo predpokladá nezávislosť atribútov, ktorými sú počty vyskytujúcich sa určitých slov v jednej dátovej vzorke. Využíva sa pri spamovacích filtroch, spracovávaní prirodzeného jazyka či klasifikovaní artiklov. Hlavná myšlienka algoritmu spočíva v nasledujúcej rovnici Bayesového pravidla.

Posterior p. likelihood class prior p.

predictor prior p

Kedže algoritmus vychádza z nezávislých atribútov, pri každom slove sa jeho vypočítaná pravdepodobnosť považuje ako nezávislý od ostatných. V jednej dátovej vzorke sa bude iterovať cez všetky slová, vynásobujúce sa navzájom ich vypočítanou pravdepodobnosťou.

Class – aktuálne vypočítavaná kategória v cykle

Word – aktuálne vypočítavané slovo v druhom vnorenom cykle

P(Word|Class) – početnosť slova v kategórii

P(Class) – početnosť kategórii

P(Word) – početnosť dištinktívneho slova

#wC – počet slov v kategórii C

#SC – počet samplov S v kategórii C

wS – aktuálne spracovávané slovo v sampli S

Pozn.: pri nevyskytujúcom sa slov v kategóriách by sa výsledná pravdepodobnosť rovnala nule. Preto sa v týchto prípadoch použila pravdepodobnosť 1-8 pre každé nevyskytujúce sa slovo.

**Stručný popis technických detailov implementácie**

Implementácia bola zrealizovaná v programovacom jazyku Python. Na Naivný Bayesovský klasifikátor sa použili knižnice **collections** na počítanie výskytu slov, **string** pre odstránenie interpukcií, pri vynásobovaní jednotlivých slov vzoriek sa výsledok často krát blížil k nule a bolo potrebné použiť funkciu z knižnice **math** funkciu logaritmus. Knižnica **random** sa použila pre náhodne zamiešanie načítaných dát (pomocou knižnice **os**) pre náhodný výber testovacej a trénovacej množiny. Na porovnanie algoritmov sa použil aj čas výpočtov pomocou knižnice **time**, ten však nebol pri výsledkoch veľmi nápomocný, pretože načítavanie dát pomocou **os** pri NB klasifikátoru bolo omnoho pomalšie ako pri SVM, ktorý používal knižnicu **sklearn**, zároveň pri načítavaní len časti dát zo súboru čítal vždy všetky dáta.

**Experimentálne vyhodnotenie**

Prvý dataset (20news-18828) ľahší na klasifikovanie pozostáva z 20 rôznych kategórií. Početnosť vzoriek v kategóriách sa pohybuje väčšinou medzi 900-1000. Niektoré kategórie sú si však príbuzné a preto sa vytvoril ďalší dataset s kombináciou len šiestich kategórií, ktoré sú najlahšie rozlíšiteľné.

Druhý dataset (ohsumed-all) pozostáva z 23 rôznych kategórií a ich početnosti sú pri každom viac rozdielne a sú uvedené v tabuľke presnejšie. Taktiež aj pre tento dataset bola vytvorená alternatíva s 6 rôznymi kategóriami.

Máme tak dokopy 4 rôzne datasety s rozdielnymi obtiažnosťami, prostredníctvom ktorých je možné zisťovať efektivitu príslušných algoritmov. Pomer trénovacej množiny s testovacou je 3:1.

Na každom datasete sa použil rôzny percentuálny(1, 2, 4, 8, 16, 32, 64, 100) počet dát na trénovanie a testovanie. Na každom z nich sa klasifikácia spustila 50 krát a presnosť predikcií sa spriemeroval. Výsledky presností sú zobrazené v nasledujúcej tabuľke.





Pri použití 100% sa vypočítal aj jeden precision, recall, f1-score avšak bez spriemerovania. Je v prílohe vysledky.csv. Z grafov je vidieť, že klasifikátor Naivného Bayesa je presnejší okrem prípadov obtiažnejších dát (ohsu-6). Pri väčšom počte kategórií s postupných nárastom dispozičných dát sa stáva presnejším NB. Je možné, že vplyv na výsledok môže mať aj rozličné vyberanie náhodných dát pri SVM a NB. Pri NB sa napr. pri 50% vyberávaní vyberá 50% z každej kategórie zvlášť, pričom vyberanie cez sklearn sa vyberá 50% zo všetkých kategórií.

**Záver**

Projekt nebol úplne dokončený a nestíhala sa podrobnejšia analýza algoritmov najmä v reporte a veľká väčšina času bola nakoniec venovaná implementácii. Presnosť a čas SVM a NB vyšli skôr naopak ako mali. Jediná presnosť ktorá sedela bola pri zložitejších datasetoch s menším počtom dát, kde SVM malo väčšiu presnosť.

**Zoznam použitej literatúry**

**[1]** [**https://www.cs.cornell.edu/people/tj/publications/joachims\_98a.pdf**](https://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf)

**[2]**[**https://www.quora.com/When-does-Naive-Bayes-perform-better-than-SVM**](https://www.quora.com/When-does-Naive-Bayes-perform-better-than-SVM)

**[3]**[**https://pdfs.semanticscholar.org/501c/fa39b2d94443ddbf4bfb7bb82c9845fa789c.pdf**](https://pdfs.semanticscholar.org/501c/fa39b2d94443ddbf4bfb7bb82c9845fa789c.pdf)

**[4]**[**http://www.ijmlc.org/papers/158-C01020-R001.pdf**](http://www.ijmlc.org/papers/158-C01020-R001.pdf)

**[5]**[**https://www.svm-tutorial.com/2014/10/svm-linear-kernel-good-text-classification/**](https://www.svm-tutorial.com/2014/10/svm-linear-kernel-good-text-classification/)

**[6]**[**https://stackoverflow.com/questions/45384185/what-is-the-difference-between-linearsvc-and-svckernel-linear/45390526**](https://stackoverflow.com/questions/45384185/what-is-the-difference-between-linearsvc-and-svckernel-linear/45390526)