Курсов проект по „Препоръчващи системи“

Иван Владимиров Иванов, Изкуствен Интелект, 23610

I. Описание на решаваната задача – Million Song Challenge

Million Song Challenge[1] е онлайн състезание за изграждането на възможно най-добрата музикална препоръчваща система. Целта е участниците да направят препоръчваща система, която да препоръча по 500 нови песни на всеки от 110 хиляди потребители, за които са дадени само половината от песните които те харесват. Предложените препоръки се оценяват като се използва критерият *Mean average precision*[2]:

$$MAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q}$$

където Q е множеството от 110-те хиляди потребители, за които са дадени само половината песни, които те харесват, а *AveP(q)* е *average precision* на препоръките дадени за *q*-тя потребител.

Като допълнителна информация състезателите разполагат с пълните данни за един милион потребители относно всички песни, които всеки един от тях харесва и редица специфични характеристики относно самите песни.

II. Предизвикателства

Количеството на данните предоставени на участниците за обучаване на техните препоръчващи системи е значимо. В компресиран формат пълният размер на данните надвишава 280GB. Въпреки, че този размер не е особено впечатляващ за индустрията, която работи с големи данни, той е смазващ за скромен лаптоп с 320GB-ов hard disk. При тази ситуация има 2 възможни решения – да се използват *Amazon Web Services*[3] (или сходна онлайн платформа за наемане на изчислителна мощ) или да се направи компромис относно пълнотата на данните върху които препоръчващата система ще бъде обучавана. Първият вариант е подходящ ако основната цел е постигането на възможно най-доброто класиране. За целта, обаче, ще бъде необходимо да се платят няколко стотин долара за наетия хардуер, дисковото пространство и генерирания мрежови трафик. Но тъй като този проект е по-скоро с академичен отколкото състезателен характер реших да направя компромис с данните – единствените данни, върху които препоръчващата система ще се обучава ще бъдат частичните данни за интересите на 110-те хиляди потребители.

Въпреки значителното орязване на използваните данни, обаче, обема на изчисленията остава значителен за един средно мощен лаптоп. За оптималното оползотворяване на ограничените хардуерни ресурси, с които разполагах, реших да напиша системата на бърз, статично типизиран, компилируем програмен език – *C++*, вместо на мощен, силно изразителен, интерпретируем език, като *Python*. С цел оползотворяване на двете ядра на лаптопа, препоръчващата система беше написана по начин, в който изчислението се разбива на независими части, всяка от които се изчислява в отделна нишка на изпълнение. От софтуерна гледна точка това изглежда адекватно, но възникна неочакван хардуерен проблем. Много-часовото 100%-ово натоварване на двете ядра на лаптопа, се оказа непосилно за охладителната

система. Компютъра постепенно започна да се нагрява все повече и повече, докато се достигна определен праг, който накара хардуера автоматично да се самоизключи.

III. Алгоритъм

Алгоритъмът, в основата на предложената препоръчваща система, е *user-based collaborative filtering*[4]. При този подход всеки потребител се представя чрез песните, които той е слушал. На високо ниво препоръките за даден потребител се изготвят, като се намират потребители със сходен „вкус“, след което предпочитанията на тези потребители се използват за препоръки. Идеята е, че ако потребител *A* има сходен „вкус“ с потребител *B* за дадена песен, то има реален шанс *A* да хареса и други песни слушани от *B*.

Има различни начини, по които може автоматично да се определи какво означава два потребителя да имат сходен „вкус“. В предложената система беше използвана мярката - *Jaccard distance*[5]. При нея всеки потребител абстрактно се представя като множеството от песните, за които се знае, че той харесва. Разстоянието между два потребителя с множества от песни *A* и *B* се дефинира като:

$$J(A,B) = \quad 1.0 \; - \; \frac{|A \cap B|}{|A \cup B|}$$

Jaccard distance-a е 0.0 ако двете множества съвпадат и 1.0 ако нямат нито един общ елемент. Като алтернативи на *Jaccard distance* биха могли да бъдат използвани и други метрики. Например потребителите могат да бъдат представени като вектори с по един елемент за всяка възможна песен. Ако потребител е слушал песен *i*, то *i*-тата компонента на неговия вектор ще бъде 1.0 а в противен случай 0.0. При това представяне сходство между два потребителя може да се сметне, като се изчисли *cosine similarity*[6] между двата вектора. Друга алтернатива е намирането на Евклидовото разстояние между тях.

Погледната по-отблизо системата прилага *k-nearest neighbor* подхода. За всеки потребител се намират *k*-те най-близки до него други потребители, като за определянето на близостта се използва мярката *Jaccard distance*. От *k*-те най-близки потребители се събират всички песни, които те са слушали, тези песни се сортират в низходящ ред по тяхната популярност (една песен е толкова популярна колкото повече различни потребители са я слушали), след което тези песни се препоръчват на въпросния потребител, ако той вече не ги е слушал. Ако препоръките получени по този начин са по-малко от необходимите 500, то те се допълват от всички възможни песни сортирани в низходящ ред по популярност до достигането на необходимата бройка.

Пристъпваме към анализ на предложения алгоритъм. В основата си за всеки потребител трябва да бъдат намерени *k*-те най-близки до него потребители. За целта трябва да се изчисли *Jaccard distance* между всяка двойка потребители. Ако общият брой на потребителите е *n*, то трябва да бъдат изчислени $O(n^2)$ *Jaccard distance-a*. Но тъй като изчисляването на *Jaccard distance* не е константна операция, а е пропорционална на средният броя песни слушани от потребител – *s*, то асимптотичната сложност става $O(s*n^2)$. Тъй като на практика *k* е константа, цената за сортиране на всички песни взети от *k*-те най-близки потребители е асимптотично несъществена в сравнение с изчисляването на *Jaccard distance* до всички потребители. Сортирането на всички песни по тяхната популярност е еднократна операция, която също е асимптотично несъществена. Или доминиращата сложност на алгоритъма си остава изчисляването на *Jaccard distance* между всяка двойка потребители – $O(s*n^2)$.

IV. Референции

[1] http://www.kaggle.com/c/msdchallenge/
[2] http://en.wikipedia.org/wiki/Information_retrieval#Mean_average_precision
[3] http://aws.amazon.com/
[4] http://en.wikipedia.org/wiki/Collaborative_filtering
[5] http://en.wikipedia.org/wiki/Jaccard_index
[6] http://en.wikipedia.org/wiki/Cosine_similarity