OD 20 TISOČ DO 20 MILIJONOV

Klemen Forstnerič, Rok Ajdnik

Reveel Technologies, Inc.
E-pošta: developers@reveel.it
URL: https://reveel.it/

POVZETEK: *Prodreti v Silicijevo dolino s kakršnokoli idejo iz Slovenije je težko, še težje pa je razviti sistem za prepoznavo več kot milijona slik. Vsak algoritem ima skromne začetke, ki jih spremljajo neuspehi. Ponavadi razvoj takšnih algoritmov temelji na izkustveni metodi. Tudi Reveel ni tukaj nobena izjema. Razvili smo algoritem, ki je deloval odlično na bazi 20000 slik, pojavili so se pa problemi s hitrostjo strežniške poizvedbe ter obdelovanjem sočasnih poizvedb, ki so onemogočali razširitev baze na milijon slik. Izkazalo se je, da smo lahko z rešitvami iz področja porazdeljenih sistemov rešili ta problem.*

1. UVOD

Pri Reveel-u [1] gradimo rešitve, ki omogočajo založnikom in oglaševalcem, narediti njihove vizualne medije interaktivne. Ena izmed rešitev je sistem za prepoznavo slik, ki omogoča, da povežemo fizični (tiskani mediji) in digitalni svet (mobilna aplikacija, spletna stran). Tiskani mediji so sami po sebi neinteraktivni, s pomočjo prepoznave slik v tiskanih medijih pa uporabnik dobi interaktivno vsebino na spletu pri čemer uporablja svojo priljubljeno mobilno napravo ali računalnik. Seveda se problem, ki ga rešujemo sedaj razlikuje od problema in ideje zaradi katere smo ustanovili podjetje. Prvotna ideja je bila, kako poenostaviti nakupovanje preko televizije. Razvijali smo, mobilno aplikacijo, ki bi uporabnikom omogočala, da med gledanjem filmov ali TV oddaj hitro in enostavno kupijo karkoli vidijo na ekranu. Tehnični problem, ki smo se ga lotili je bil, kako z uporabo senzorjev v pametnem telefonu odkriti, kaj uporabnik gleda na televiziji, platnu ali monitorju.

2. ZAČETKI

Med raziskovanjem smo ugotovili, da lahko problem rešimo na tri različne načine. Od uporabnika bi lahko zahtevali, da sam poišče in izbere vsebino, katero gleda. Tako bi problem rešili izključno s standardnimi rešitvami uporabniških vmesnikov. Takšna rešitev bi zahtevala najmanj razvojnega časa, ampak bi z vedno več vsebine v sistemu postala mučna za uporabnika. Naslednja rešitev, ki smo jo raziskali je uporaba avdio prepoznave, ki bi omogočala prepoznavo video vsebin preko njihovih avdio zapisov. Prednost tega pristopa je manjša baza začetnih podatkov, saj je avdio zapis videa manjši kot njegov video zapis. Druga prednost z uporabniške izkušnje je lažja uporaba, saj lahko mobilno napravo držimo v udobnem položaju medtem ko prepoznava avdio. Pri video prepoznavi moramo

kamero mobilne naprave usmeriti proti videu, da lahko le-ta uspešno izvede prepoznavo. Zadnja rešitev pa je video prepoznava. Glavna prednost tega pristopa je, da ima veliko večji potencial za prihodnost, saj ni omejen samo na video vsebino ampak lahko tehnologija razpozna tudi revije in ostale tiskane medije. Medtem ko imata oba pristopa, avdio in video razpoznava, svoje prednosti in slabosti, je bil glavni faktor zakaj smo se odločili za video razpoznavo, da bi se odtujili od naše konkurence.

2.1 Digitalni vodni žig

Prvotna ideja je bila uporaba digitalnih vodnih žigov - “digital watermark” [2] v videih. V video smo zakodirali digitalne vodne žige, s pomočjo kamere na mobilni napravi zajeli sliko, dekodirali digitalni vodni žig in tako prepoznali video. Digitalni vodni žig je postopek, kjer v celotno sliko zapišemo podatke. Pri tem je pomembno, da je popačenje neopazno človeškemu očesu, vendar vidno dekodirnemu algoritmu, ki iz slike izračuna N bitno kodo, ki unikatno identificira vsebino. Seveda se digitalnega vodnega žiga ne zaznava na originalni sliki, ampak na sliki, ki je zajeta s kamero mobilne naprave in vsebuje to originalno sliko. Zato mora algoritem imeti visoko toleranco na šum. Visoko toleranco na šum je možno doseči z uporabo visoke redundance podatkov med kodiranjem digitalnega vodnega žiga, ampak to pa pomeni, da smo omejeni s količino podatkov, ki se lahko zakodirajo v sliko. Vnašanje digitalnega vodnega žiga v video vsebine poteka tako da skozi vse slike videa zakodiramo *N* bitno kodo, kjer prvih *X* bitov unikatno identificira video vsebino, drugih *Y* bitov pa časovno lokacijo v vsebini. Ob zajemanju slike videa preko mobilne naprave se najprej izvede registracija robov televizije ali monitorja, perspektivna korekcija zajete slike in nato ekstrakcija podatkov digitalnega vodnega žiga. Naš prvi prototip je bil algoritem opisan v članku Meerwald et.al. [3], s katerim smo lahko zakodirali 56 bitov v posamezno sliko, kjer je prvih 32 bitov identificiralo vsebino, preostalih 24 bitov pa je predstavljalo časovno lokacijo v vsebini. Algoritem za vstavljanje digitalnega vodnega žiga uporablja diskretno valjčno transformacijo s katero ob uporabi “spread spectrum” kodiranja zakodira podatke v originalno sliko. Ob testiranju smo ugotovili, da je učinkovitost algoritma zelo odvisna od uspešnosti registracije robov televizije ali monitorja. Registracijo robov smo izvajali tako, da smo uporabljali kombinacijo več algoritmov. Najprej smo izvedli Canny za detekcijo robov na sliki, Hough transformacijo za iskanje premic, nato pa smo poiskali najbolj primerna oglišča glede na kriterij, da morajo imeti presečišča notranji kot okoli 90 stopinj. Preden bi lahko nadaljevali razvoj in raziskave smo ugotovili, da naša rešitev ni najbolj primerna za trg v katerem smo želeli lansirati našo rešitev. Rešitev zahteva spreminjanje video vsebine, saj moramo vanjo vstaviti digitalni vodni žig, kar pa bi pomenilo, da bi morali zakupiti licenco za vsako vsebino v katero bi želeli vstaviti digitalni vodni žig. S stališča marketinga in prodaje se je ta pristop izkazal za neizvedljivega oziroma zelo dragega.

2.2 Značilke, gručenje in uteži

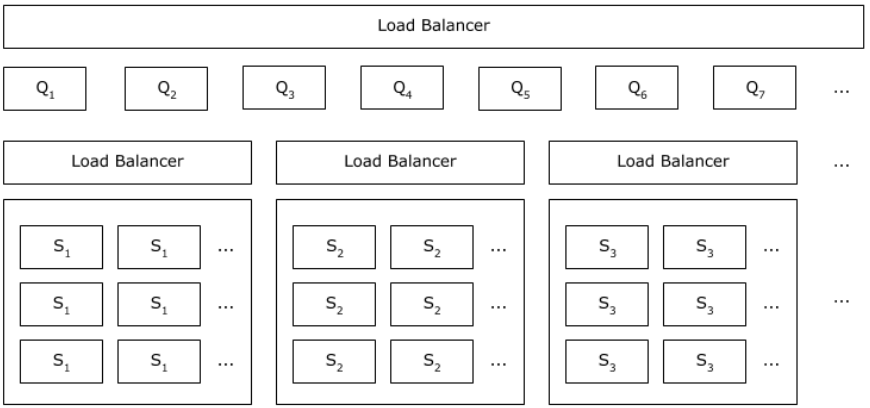
Zaradi zahtev trga smo bili primorani spremeniti naš pristop in ugotoviti kako prepoznati vsebino ne da bi jo spreminjali. Med raziskovanjem smo naleteli na področje algoritmov s skupnim imenom “Content-based Image Retrieval” ali s kratico CBIR [4]. Ti algoritmi

delujejo tako, da najprej ustvarijo bazo slik in nato omogočajo iskanje nad to bazo. Pri iskanju najdejo v bazi sliko, ki se najbolj ujema z iskano sliko. Pri implementaciji naše CBIR rešitve smo se odločili uporabiti pristop, kjer smo iz slik izločili značilke in nato uporabili te značilke pri gradnji baze in iskanju. Pri izbiri značilk smo iskali takšne, ki bodo robustne na popačenja in šum, ki nastanejo na sliki, ko iskano sliko zajamemo s kamero mobilne naprave. Druga zahteva, ki smo jo imeli zaradi omejitev prepoznavanja videov pa je bila hitrost algoritma. Celotna poizvedba se mora na strežniku izvesti hitro, da zagotovimo, da bo video vsebina prepoznana v doglednem času za uporabnika. Zaradi teh omejitev smo se odločili za uporabo značilk ORB [5]. Algoritem je dovolj robusten na transformacije in šum ter kljub vsemu zelo hiter. Predvsem pa nas je prepričalo dejstvo, da algoritem ni licenciran in se ga lahko prosto uporablja. Nato je nastopil drug problem in sicer kako uspešno izvesti iskanje nad bazo značilk. Če poenostavimo, imamo dve množici značilk, prvo množico predstavljajo značilke, katere smo izračunali iz vseh slik, ki jih imamo v bazi, drugo množico pa predstavljajo značilke, katere smo izračunali iz slike s katero izvajamo iskanje. V praksi druga množica vsebuje med 1000 in 2000 značilk, prva množica pa okoli 28 milijonov značilk. Hitro se je izkazalo, da je enostavno linearno iskanje skozi bazo neizvedljivo, predvsem zaradi časovnih omejitev, ki jih imamo pri poizvedbah. Ob raziskovanju rešitev, ki zadostujejo vsem omejitvam smo naleteli na skupino algoritmov FLANN [6], ki omogočajo hitro iskanje skozi velike baze binarnih ali vektorskih podatkov. Značilke ORB so v binarnem formatu, zato smo se odločili za uporabo “Hierarchical Clustering” dreves [7]. S pomočjo FLANN pristopa smo lahko hitro poiskali vse značilke, ki se najbolj ujemajo z iskano množico značilk. Pri tem pa smo naleteli na drug problem, ki nastane pri zajemu slike z mobilno napravo - da se večina značilk originalne slike spremeni kljub robustnosti ORB algoritma. Ta problem nas je privedel do ideje o implementaciji gručenja, kjer smo vse značilke posplošili na nek vnaprej definiran nabor značilk. Ta pristop je pospešil iskanje skozi indeks FLANN in rešil problem značilk, ki so popačene zaradi šuma in artefaktov. Po več mesecih poskusov in neuspehov smo našli članek Sivic et. al. [8], ki nas je usmeril v pravo smer in postavil temelje trenutnem algoritmu. Članek opisuje podoben pristop, kot smo ga sami ubrali vendar predstavi nekaj ključnih rešitev, ki doprinesejo k delujočemu sistemu za razpoznavo. Članek opisuje indeks FLANN, ki je zgrajen iz milijona vnaprej izbranih značilk, s katerim se zgradi podatkovna baza. Med iskanjem se z uporabo indeksa FLANN iz baze poišče podobne slike, ki se nato s TF-IDF [9] utežmi zmanjša na manjšo množico. Nad to množico se nato še linearno izvede geometrična verifikacija ključnih točk z algoritmom RANSAC [10]. S tem algoritmom lahko v manj kot 300 milisekundah izvedemo iskanje skozi bazo 20000 do 50000 slik, odvisno od vsebine slik. Ta sistem za prepoznavo slik smo uporabljali v našem produktu dobro leto in pol, dokler se niso pojavile prepreke zaradi katerih smo morali nadgraditi naš sistem.

3. NADGRADNJA SISTEMA

Sistem, ki je bil načrtovan in implementiran pod velikim časovnim pritiskom zaradi pridobivanja investicije, je sčasoma začel kazati znake degradacije. Sprva smo prepoznavali samo video vsebine, tekom delovanja pa smo v sistem vnašali vse več slik strani revij in ostalih tiskanih medijev, ki pa imajo drugačne karakteristike kot video.

Pojavili so se problemi z dostopnostjo storitve v času povečanega obiska uporabnikov ter v primeru dodajanja nove vsebine v sistem. Časi obdelave posameznega zahtevka so se iz manj kot 300 milisekund povišali na 5 in več sekund, kar je za uporabnika nesprejemljivo. Potrebno je bilo drastično zmanjšati iskalni prostor po katerem išče strežnik, obenem pa obdržati vse obstoječe slike v sistemu. Ena izmed rešitev za ta problem, je implementacija distribuiranega sistema, pri katerem vsi strežniki nimajo dostopnih vseh podatkov v sistemu. Ločili smo strežnik na dva dela, in sicer na strežnik, ki sprejema poizvedbe od uporabnika - "query", ter na strežnik, ki hrani trenutno dostopne slike - "storage". Strežnikov "storage" je običajno več, in vsak ima le del vseh slik trenutno v sistemu. S tem pristopom učinkovito zmanjšamo čas, ki je potreben za obdelavo poizvedbe. Ko dobimo poizvedbo, se ta pošlje na "query", ki dalje paralelno posreduje to poizvedbo s pomočjo protokola RPC [11] na vse "storage" strežnike. Vsak izmed "storage" strežnikov vrne identifikator najboljše slike, ki jo ima - v primeru da ta obstaja - ter numerično utež, ki ponazarja, kako prepričan je, da je ta slika prava. Strežnik "query" nato rangira rezultate in vrne uporabniku najboljšega. Tak pristop se, sodeč po testih obnese bistveno bolje od starega pristopa. Vseeno pa ima še en problem. V primeru, da zaradi katerega koli razloga en strežnik postane nedostopen, uporabniki več ne morejo dostopati do dela slik. Rešitev je, da vsakega izmed "storage" strežnikov še dodatno N-krat repliciramo s čimer povečamo redundanco za ta del slik v primeru, da katerikoli od teh strežnikov postane nedostopen. S tem pristopom resda učinkovito rešimo problem z nedostopnostjo, potrebno pa je poudariti, da pride s tem do dodatnih problemov. Pomembno je namreč, da se vsi strežniki, ki hranijo določen del slik, strinjajo o tem kakšen je ta del, ki ga hranijo. Pridemo torej do problema konsenza, ki je znan problem v distribuiranih sistemih. Obstaja ogromno algoritmov, ki rešujejo ta problem, najbolj znan med njimi je Paxos [12], ki pa je znan po tem, da ga je v realnem svetu zelo težavno implementirati. Zaradi tega dejstva, smo se odločili, da uporabimo algoritem Raft [13], ki je bil načrtovan tako, da je enostaven za implementacijo. Arhitekturno gledano smo naš sistem posodobili iz monolitičnega, ki je vseboval funkcionalnost "query" in "storage" v arhitekturo, ki je prikazana na Slika 1.



Slika 1: Arhitektura nadgrajenega sistema za prepoznavo slik, kjer so Q ti. query strežniki in S ti. storage strežniki.

4. PRIHODNJI IZZIVI

Kljub vsem izboljšavam imamo še veliko za postoriti, preden bo naš sistem zmogel prepoznati 20 milijonov slik. Od razvoja dodatnih funkcij, ki bodo sistemu razširile zmogljivost, do izboljšav že obstoječih algoritmov. V nadaljevanju bomo predstavili nekaj problemov, ki jih še nismo rešili in opisali naše pristope k njihovim rešitvam. Glede na to, da se Reveel vseskozi prilagaja različnim trgom v katere vstopamo, se bodo zahteve sistema za prepoznavo nedvomno spremenile, kar pomeni, da se bomo v prihodnosti srečali s problemi, na katere še nismo pomislili.

4.1 Predpomnjenje

Naš sistem v povprečju posamezen zahtevek obdela v 300 milisekundah. Čeprav je to zelo hitro, je smiselno, da vsebino, do katere uporabniki pogosteje dostopajo, shranimo v predpomnilniku, saj lahko na tak način do nje dostopamo še hitreje in zmanjšamo obremenitev strežnikov. Načinov kako zasnovati predpomnilnik je več in ena izmed teh je uporaba zgoščevalnih funkcij - “hashing functions”. V našem primeru imamo namen uporabiti percepcijske zgoščevalne funkcije - “perceptual hashing functions” [14], ki so zmožne zmanjšati dimenzionalnost slik na nekaj zlogov in vseeno ohraniti osnovne informacije o sliki. To pomeni, da si bosta rezultata zgoščevalne funkcije dveh podobnih slik prav tako podobna. Z uporabo zgoščevalnih funkcij in dreves “Hierarchical Clustering” lahko nato zgradimo predpomnilnik, ki bo lahko hitro poiskal, če se kakšna poizvedba ujema s prejšnjimi poizvedbami in namesto, da ponovimo iskalni algoritem lahko vnaprej vrnemo rezultat, ki se je izračunal pri prejšnji poizvedbi.

4.2 Geometrična verifikacija

Analiza korakov našega algoritma za prepoznavo nam je pokazala, da se vsaj 50% časa ob vsaki poizvedbi porabi pri geometrični verifikaciji ključnih točk, ki jo rešujemo z algoritmom RANSAC. To pomeni, da je izboljšava tega koraka ključna za celostno pohitritev algoritma. Od leta 1980, ko je izšel članek za RANSAC, se je pojavilo ogromno izboljšav, ena izmed boljših trenutno je Graph-Cut RANSAC [15]. GC-RANSAC uporablja principe iz teorije grafov in v enem izmed korakov iz ključnih točk zgradi graf pretočnega omrežja oz. “flow network” nad katerim nato izvede reze s čimer pohitri iskanje vključenih točk oz. “inliers”. Sodeč po članku, pričakujemo približno 40% pohitritev na račun zamenjave algoritma.

5. ZAKLJUČEK

Štiri leta nazaj smo začeli z enostavno idejo, kako poenostaviti nakupovanje gledalcem malih in velikih ekranov. Tekom let se je ta ideja spremenila mnogokrat in z njo se je tudi spremenila naša rešitev. Dokaj hitro smo ugotovili, da je neuspeh naš najboljši učitelj in da bomo do prave rešitve prišli samo, če smo pripravljeni mnogokrat spodleteti.

LITERATURA

1. *https://reveel.it/*
   Reveel - Make all your media interactive so you can invite audiences to explore, learn and buy.

2. L. K. Saini, V. Shirvastava (2014), A Survey of Digital Watermarking Techniques and its Applications, *International Journal of Computer Science Trends and Technology*, vol. 2, str. 70-73.

3. P. Meerwald, A. Uhl (2010), Watermark Detection for Video Bookmarking Using Mobile Phone Camera, *Communications and Multimedia Security, Lecture Notes in Computer Science*, vol. 6109, str. 64-74.

4. W. Zhou, H. Li, Q. Tian (2017), Recent Advance in Content-based Image Retrieval: A Literature Survey, *https://arxiv.org/abs/1706.06064v2*.

5. E. Rublee, V. Rabaud, K. Konolige, G. Bradski (2011), ORB: an efficient alternative to SIFT or SURF, *Proceedings of the IEEE International Conference on Computer Vision*, str. 2564-2571.

6. M. Muja, D. G. Lowe (2009), Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration, *International Conference on Computer Vision Theory and Applications*.

7. M. Muja, D. G. Lowe (2012), Fast Matching of Binary Features, *Conference on Computer and Robot Vision*.

8. J. Sivic, A. Zisserman (2003), Video Google: a text retrieval approach to object matching in videos, *Proceedings of the Ninth IEEE International Conference on Computer Vision*, vol. 2.

9. H. Wu, R. Luk, K. Wong, K. Kwok (2008), Interpreting TF-IDF term weights as making relevance decisions, *ACM Transactions on Information Systems*, vol. 26.

10. S. Choi, T. Kim, W. Yu (2009), Performance Evaluation of RANSAC Family, *Proceedings of the British Machine Vision Conference*, vol. 24, str. 1-12.

11. *https://tools.ietf.org/html/rfc5531*
    RFC 5531 - RPC: Remote Procedure Call Protocol Specification Version 2

12. L. Lamport (1998), The Part-Time Parliament, *ACM Transactions on Computer Systems 16*, vol. 2, str. 133-169.

13. D. Ongaro, J. Ousterhout (2014), In search of an understandable consensus algorithm, *Proceeding USENIX ATC'14 Proceedings of the 2014 USENIX conference on USENIX Annual Technical Conference*, str. 305-320.

14. C. Zauner (2010), Implementation and Benchmarking of Perceptual Image Hash Functions, *Master's thesis, Upper Austria University of Applied Sciences, Hagenberg Campus*.

15. D. Barath, J. Matas (2017), Graph-Cut RANSAC, *https://arxiv.org/abs/1706.00984*.