Plataforma interactiva per a la comparació de detectors de text

Jaume Delclòs Coll

Resum– Els desenvolupaments en inteligència artificial acostumen a quedar amagats del públic general fins que no arriben a productes comercials. Ens proposem de fer arribar l'estat de l'art en detectors de text a un públic més ampli, fora del context acadèmic, i crear un entorn en que les reaccions del públic puguin potencialment entrenar els sistemes d'aprenentatge computacional. Ho fem a través d'una plataforma web oberta a tothom, on qualsevol persona pot posar a prova els detectors amb les seves imatges i valorar-ne els resultats.

Paraules clau– Visió per computador, web, RRC, Robust Reading Competition.

Abstract– The new developments in artificial intelligence tend to stay hidden from the general public until they are put to use in comercial products. We aim to bring the state of the art in text detection to a broader public, outside of academia, and to create an environment where the reactions from people could potentially be used to train the machine learning systems. We do so through a web-based platform open to everybody, where anybody can test the detectors against their own pictures and rate the outputs.

Keywords– Computer Vision, web, RRC, Robust Reading Competition.

1 Introducció

El reconeixement de text en imatges és un tema de gran importància dins el camp de la intel·ligència artificial.

Cada any apareixen nous mètodes atacant variants del problema, que es poden comparar amb diferents mètriques. Concretament, la Robust Reading Competition de l'ICDAR [1] és una competició biennal on es comparen diferents mètodes de detecció i reconeixement de text. Kaggle és el portal de competicions de sistemes d'aprenentatge computacional més important a escala mundial, però sense estar centrat només en text. Si bé té competicions en moltes categories d'Intel·ligència Artificial, en detecció i reconeixement de text, la més prestigiosa és la RRC de l'ICDAR.

El programari en qüestió però, no és sempre públic, i quan ho és, no acostuma fàcil de fer servir per al públic general. És a dir, els avanços i resultats en el camp de l'extracció de text no són accessibles fora del context acadèmic o de recerca.

L'objectiu principal del projecte doncs, és apropar els nous desenvolupaments en tecnologia de reconeixement i extracció de text al públic sense perfil tècnic. El medi per a fer-ho serà crear una plataforma Web pública, que permeti a qualsevol persona comparar un nombre de mètodes d'extracció de text amb les seves pròpies imatges, de manera senzilla. A més, aquesta plataforma podrà servir, tot i que queda fora de l'abast del Treball de Fi de Grau, com a la primera pedra per a un sistema en que els sistemes d'aprenentatge computacional s'entrenessin amb les imatges enviades pels usuaris i les seves corresponents valoracions (weak-anotations), i crear un sistema d'entrenament de Human-in-the-loop[2][3].

Prenem inspiració de demos amb alta usabilitat com les del Watson Visual Recognition d'IBM o el Cloud Vision API de Google (figura 1).

Aquestes demos serveixen com a eina de màrqueting a aquestes grans empreses per a vendre el seu producte. Aquest projecte, per contra, no intenta vendre cap producte propi, sinó que ensenya els resultats d'algorismes de tercers de domini públic, que participen en competicions. Però el funcionament serà similar: l'usuari podrà triar una imatge del seu ordinador o mòbil, i veurà els resultats dels algorismes.

Aquesta plataforma tindrà valor per als investigadors perquè els donarà publicitat als algorismes. Farà que els resultats de la seva feina passin de la taula estadística que en descriu la precisió a una cosa més tangible, més propera, i més visible: una plataforma on tothom pot veure fàcilment com de bé fa la feina.

• E-mail de contacte: jaume@delclos.com
• Menció realitzada: Computació
• Treball tutoritzat per: Dimosthenis Karatzas (CVC)
• Curs 2019/2020

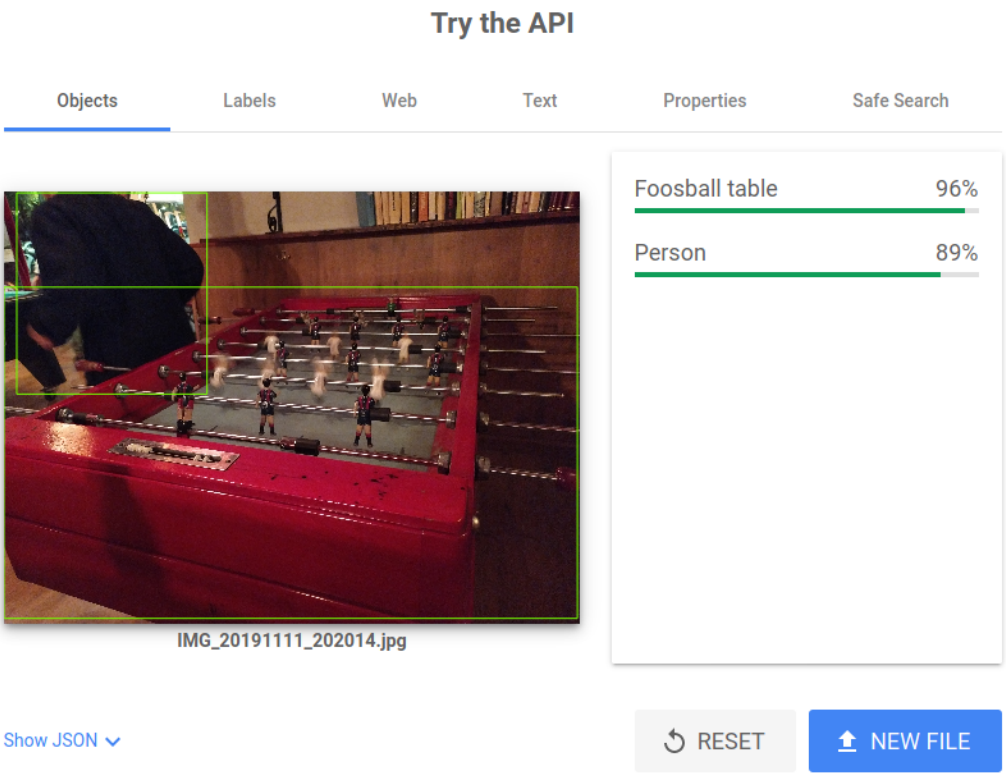

Fig. 1: Demo de Google

2 Objectius del projecte

1. Implementar el front end i el back end del servei web.

   S'ha de programar tant el codi del servidor web, com la part que s'executarà al navegador de l'usuari.

2. Definir un Remote Procedure Call perquè tercers puguin afegir algorismes a la plataforma.

   Hem de crear un estàndard que funcioni a través d'Internet. També l'utilitzarem per als nostres propis algorismes.

3. Implementar un mínim de tres algorismes de processament d'imatges.

   S'escolliran tres algorismes representatius d'entre els participants de la Robust Reading Competition que tinguin el codi publicat, i s'adaptaran per a funcionar dins el nostre sistema.

4. Fer proves per comprovar el funcionament correcte del sistema i analitzar les dades dels usuaris.

   Es faran proves amb usuaris del públic general per a demostrar que la plataforma és usable. Després s'analitzaran les dades obtingudes.

3 Estat de l'art

Per a entrenar sistemes d'aprenentatge computacional es fan servir conjunts de dades anotades. En el cas del reconeixement de text, imatges anotades amb la posició i contingut del text. Com que anotar de manera precisa grans col·leccions d'imatges és una feina laboriosa, aquests conjunts de dades acostumen a ser o bé massa petits, o estar formats d'imatges sintètiques generades per ordinador. És a dir, no són fotografies reals on hi aparegui text amb les corresponents anotacions, sinó que, o bé són imatges generades en entorns completament virtuals, o bé són fotografies a les quals s'ha afegit text per ordinador. És per això que una plataforma on es recollissin imatges per part d'usuaris a gran escala podria ajudar a tirar endavant l'estat de l'art significativament.

El primer pas serà veure quins mètodes podem escollir per posar a la disposició del usuaris a la nostra plataforma. A la taula 1 hi ha alguns dels mètodes de la competició RRC de 2019 que han publicat el seu codi i models entrenats, i per tant són opcions a adaptar per a la nostra plataforma:

L'elecció dels mètodes en concret es farà avaluant la facilitat d'implementació. En altres paraules: inicialment s'escolliran els mètodes amb les instruccions d'instal·lació i execució més clares.

Després d'una petita anàlisi d'alt nivell, veiem que tots estan escrits en python, i la gran majoria utilitzant la biblioteca PyTorch. PyTorch gairebé sempre s'entrena en GPUs amb l'ajuda de CUDA, però permet executar els models ja entrenats en CPU. Així doncs, no ens caldrà que el nostre servidor tingui una targeta gràfica.

Fent proves també hem vist que si les imatges tenen una resolució alta, els algorismes tendeixen a consumir molta RAM. Caldrà fer més proves per decidir quanta RAM necessitem realment, i quines limitacions de resolució hem de posar.

4 Metodologia i tecnologies

El projecte es desenvoluparà a base de *sprints* setmanals seguint les tasques descrites a la secció de Planificació. Cada setmana s'avançarà en les tasques adients seguint l'ordre del projecte.

TAULA 1: MÈTODES AMB CODI PÚBLIC

Challenge	Data	Repositori
MLT/LSVT	2019-05-30	`https://github.com/STVIR/PMTD` [4]
MLT/LSVT	2019-06-02	`https://github.com/facebookresearch/Detectron` [5]
LSVT/ArT	2019-04-30	`https://github.com/facebookresearch/maskrcnn-benchmark` [6]
LSVT/ArT	2019-04-30	`https://github.com/open-mmlab/mmdetection` [7]
LSVT	2019-04-29	`https://github.com/huoyijie/AdvancedEAST` [8]
ArT	2019-04-25	`https://github.com/whai362/PSENet` [9]
ArT	2019-04-28	`https://github.com/princewang1994/TextSnake.pytorch` [10] + `https://github.com/whai362/PSENet` [11]
SROIE	2019-04-22	`https://github.com/lvpengyuan/corner` [12] `https://github.com/argman/EAST` [13]
SROIE	2019-04-22	`https://github.com/CharlesShang/DCNv2` [14] + PSENet [11]
SROIE	2019-04-21	`https://github.com/tianzhi0549/CTPN` [15]
SROIE	2019-04-22	`https://github.com/loveorchids/sroie2019` [16]
SROIE	2019-04-17	`https://github.com/eragonruan/text-detection-ctpn` [15]
SROIE	2019-04-17	`https://github.com/AlexeyAB/darknet`
ReCTS	2019-04-30	`https://github.com/clovaai/deep-text-recognition-benchmark` [17]

5 PLANIFICACIÓ INICIAL

Les tasques de la fase 1 del projecte són les següents:

- Elecció dels mètodes (15 de març)
  - Entendre quines dades en podem extreure, i com les podem presentar
  - Fer proves amb cada un per assegurar-ne la viabilitat
- Creació d'un back end i front end mínims (22 de març)
- Integració amb els nuclis (12 d'abril)
- Documentació del protocol (19 d'abril)
- Preparació d'un servidor públic (26 d'abril)

Com a segona fase les tasques són:

- Polir la interfície (3 de maig)
- Assegurar el rendiment correcte amb múltiples usuaris (10 de maig)
- Afegir més nuclis (17 de maig)

La documentació de tot el procés es farà en paral·lel, és a dir, cada tasca portarà la seva documentació.

El testeig del projecte serà manual, i es realitzarà després de cada tasca. En principi no es preveu l'ús de *unit tests* ni altres tests automàtics, ja que és un esforç redundant al testeig manual continu que es farà durant el desenvolupament. Un cop acabat el projecte, es demanarà a terceres persones de manera informal que provin l'aplicació per comprovarne la usabilitat i detectar possibles errors.

La planificació inicial era la que es veu a la figura 12 de l'apèndix.

El desenvolupament del projecte es va endarrerir greument per dos motius. En primer lloc, a causa de l'alarma del coronavirus, es van començar diversos projectes urgents a la meva feina que es van sumar a la càrrega de feina normal i em van mantenir molt ocupat durant dues setmanes i fora del TFG.

En segon lloc, la planificació també es va veure endarrerida per una baixa de salut entre finals de març i principis d'abril.

Aquest endarreriment però, no va posar en perill el projecte. És a dir, continuava sent viable amb un ajust de planificació, representat a la figura 13 de l'apèndix.

6 ESPECIFICACIONS

6.1 Requeriments

Els requeriments per al nucli tecnològic del projecte són els següents:

1. L'aplicació ha de ser una aplicació web, accessible des de qualsevol mòbil o ordinador amb un navegador.
2. S'han de poder pujar imatges per a ser processades.
3. Hi ha d'haver 3 o més mètodes de processament d'imatges.
   1) L'usuari ha de poder comparar els mètodes entre ells.
   2) Hi ha d'haver una manera documentada per afegir nous mètodes.
4. L'usuari ha de poder permetre o no que la seva imatge es desi per a ser utilitzada en entrenament de mètodes futurs.

6.2 Cicle de procés bàsic

Com veiem al diagrama de seqüència de la figura 2, el flux que seguirà l'usuari serà el següent:

1. L'usuari arriba a la pàgina inicial, on pot veure la descripció del lloc web i el formulari amb les caselles de *Termes i Condicions* i permís per a desar la imatge.
2. L'usuari accepta els termes i envia una imatge a través del formulari.
3. El servidor web rep la imatge i la desa a la seva memòria.
4. El servidor web demana al servidor de processament d'imatges que comenci a processar la imatge.
5. El servidor web notifica l'usuari que el procés s'ha iniciat.
6. El servidor d'imatges acaba de processar la imatge i ho notifica a través de WebSockets.
7. El navegador rep la notificació i descarrega els resultats.

8. L'usuari pot veure els resultats al navegador.

6.3 Tractament de dades i privacitat

El nostre projecte consisteix en un servei que tracta amb dades de persones terceres, i com a tal té un conjunt de qüestions a tractar pel que fa a la llei de protecció de dades i la privacitat dels usuaris. Per a minimitzar els possibles problemes, no es permetrà als usuaris publicar les imatges que pugin al sistema a la nostra plataforma. És a dir, pujaran les imatges i en veuran ells mateixos els resultats. Si els volen compartir, hauran de descarregar els resultats i publicar-los ells mateixos. A més, les imatges dels usuaris es desaran per defecte només en emmagatzematge temporal de curt termini.

El que sí que es permetrà a l'usuari és expressar que vol permetre que la seva imatge s'utilitzi per a millorar els algoritmes d'intel·ligència artificial. En aquest cas, la imatge no podrà contenir informació que es pugui considerar confidencial, dades personalment identificables, ni estar protegida per drets d'autor. En aquest cas, la imatge es podrà desar en una base de dades per al seu ús a llarg termini.

Els usuaris accediran a la plataforma de manera anònima, i per tant només en desarem l'identificador de sessió a la memòria cau i les dades de les imatges en processament.

7 DESENVOLUPAMENT

7.1 Tecnologies

A nivell tècnic, la plataforma consistirà de tres parts fonamentals. En primer lloc, el *front end*, entès com la interfície que s'executa directament al navegador de l'usuari. En segon lloc, la part web del *back end*, encarregada de gestionar les peticions HTTP. En tercer lloc, els diferents nuclis de visió per computador, encarregats de processar les imatges. Vegeu la figura 3.

Per al front end s'utilitzarà HTML5, i Javascript en els llocs on permeti enriquir l'experiencia de l'usuari. En principi es descarta utilitzar frameworks més complexos com podrien ser React, Angular o Vue.js, ja que volem una interfície senzilla amb poques interaccions: un formulari d'entrada, una pàgina de resultats, etc.

Per al back end web s'utilitzarà el llenguatge de programació Elixir, amb el web framework Phoenix. El model d'actors que utilitza Elixir facilitarà la feina a l'hora de treballar amb processos de llarga durada com pot ser el processament d'imatges en alguns casos. Cal dir que podrien servir també altres frameworks com Laravel, Ruby on Rails, Django o Express.js. Tots compleixen els requisits del projecte i per tant l'elecció ve donada en gran part per la comoditat que suposa el llenguatge Elixir.

Si bé la majoria de mètodes públics al nostre abast utilitzen Python com a llenguatge de programació, caldrà aïllarlos amb les versions de les biblioteques necessàries en cada cas al nostre servidor. Per això es preveu l'ús d'entorns virtuals utilitzant Poetry o Conda. L'API entre el backend i els mètodes utilitzarà el mateix protocol d'RPC estàndard per a tots els mètodes, siguin propis o de tercers.

Tot el back end funcionarà a sobre d'un servidor Debian GNU/Linux. Les especificacions del maquinari s'ajustaran un cop hi hagi un prototip en funcionament.

Inicialment el projecte s'executarà a sobre el mateix PC de treball, i un cop estigui llest es portarà al servidor de producció.

7.2 Protocol de comunicació

Cada mètode de processament d'imatge és accessible des d'un servidor HTTP. A un URL definida, ha d'acceptar una petició POST amb un cos JSON, de la següent forma:

```
{
    filename: "image.png",
    image: "<imatge en base64>"
}
```

La imatge sempre vindrà en format PNG, ja que la preprocessem al servidor web. En cas que sigui més gran de 1000px en alçada o amplada, la redimensionem mantenint la proporció.

La resposta de la petició ha de ser altre cop en format JSON, amb els paràmetres:

```
{
    mime: "image/png",
    image: "<imatge en base64>"
}
```

La imatge pot estar en qualsevol format que puguin acceptar els navegadors. Amb els mètodes que hi ha implementats, fem servir `image/png` i `image/jpeg`.

Per a afegir o modificar un mètode, hi ha un tauler de control a l'aplicació web que veiem a la figura 4.

Ara per ara, l'única versió del protocol és la 1 (la que està explicada a sobre).

7.3 Mètodes

Cada mètode està embolcallat amb un servidor web Flask molt senzill escrit en Python. Els mètodes en si també estan escrits tots tres en python, però no tindrien per què estar-ho, ja que s'executen en processos separats.

El procés de PixelLink s'executa en Python 2, mentre que els altres dos s'executen en Python 3. Els servidors flask en canvi, són tots Python 3.

Com a exemple, a l'apèndix A.2 podem veure el servidor web que embolcalla l'EAST.

La gràcia però d'aquest sistema, és que un proveïdor tercer no tindria per què fer servir aquest codi, sinó que podria fer servir qualsevol sistema per rebre les peticions i enviar les respostes: un script en PHP, un servidor web de Ruby on Rails, o qualsevol altre entorn de programació web. És una interfície programàtica molt senzilla i universal.

7.4 Disseny intern

El cor de l'aplicació és un servidor escrit en el llenguatge de programació Elixir, que s'executa sobre la màquina virtual BEAM d'Erlang OTP. El que més ens interessa d'aquest sistema és el seu paradigma de concurrència. Sense que hàgim de preocupar-nos de gestionar *threads* o funcions asíncrones, el sistema BEAM utilitza un sistema propi de processos virtuals amb missatges entre ells per a fer les tasques de manera paralel·la i sense bloquejos. Aquests processos no són processos d'Unix tradicionals, sinó que són

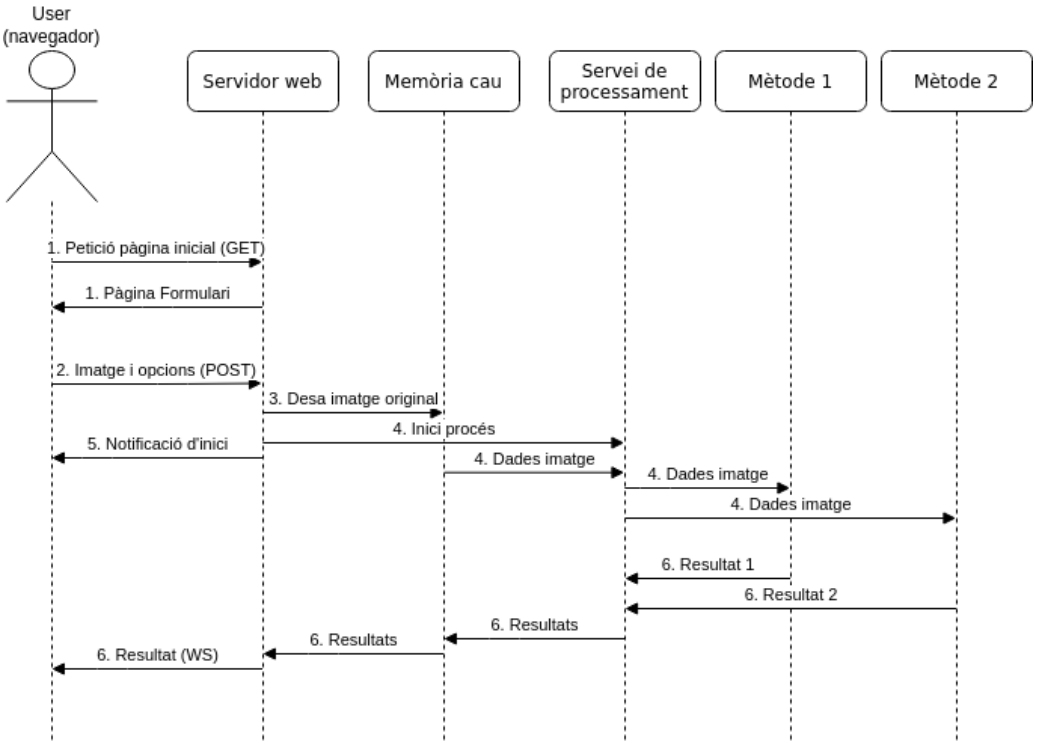

Fig. 2: Diagrama de seqüència

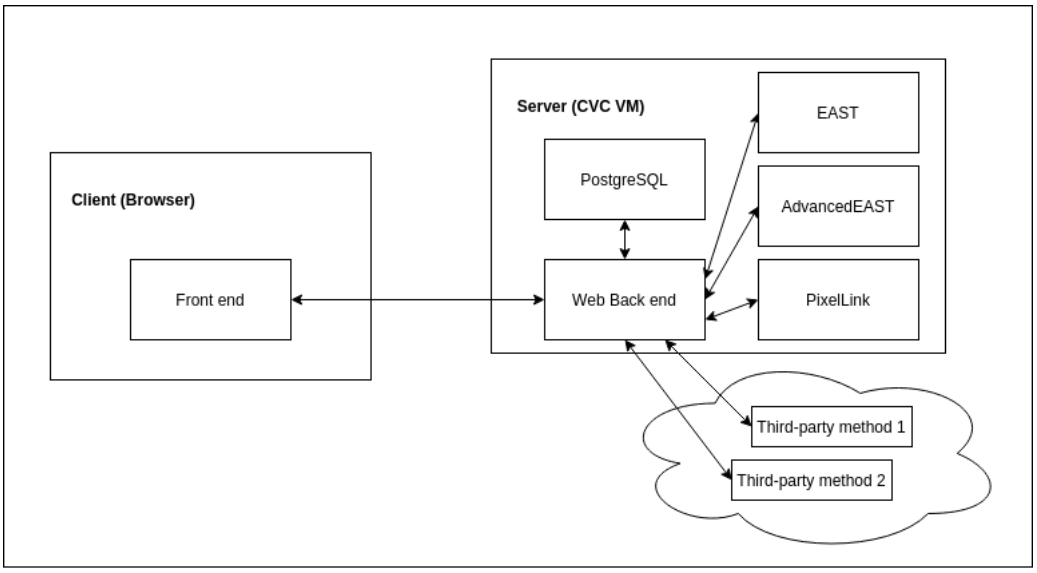

Fig. 3: Diagrama de l'estructura

Listing Methods

Name	Url	Version	
PixelLink	http://127.0.0.1:4012/process	1	Show Edit Delete
AdvancedEast	http://127.0.0.1:4011/process	1	Show Edit Delete
EAST	http://127.0.0.1:4010/process	1	Show Edit Delete

New Method

Fig. 4: Mètodes

molt més lleugers i no suposen un problema de rendiment. Ben al contrari, permeten repartir millor la feina entre els diferents threads d'execució del processador, i una millor resistència en cas d'errors. Si un d'aquests processos falla, es reinicia automàticament sense fer fallar el sistema. D'aquesta manera, un descuit del programador que podria ser fatal en un altre llenguatge de programació pot ser irrellevant en Elixir.

La biblioteca de programació web que utilitzem s'anomena Phoenix, i està organitzada seguint el patró de disseny Model View Controller. En el nostre cas, la part més important del Model és el servei de processament d'imatges. Les nostres Views són esquelets d'HTML on introduïm les dades corresponents a cada pàgina. I finalment, els Controllers tenen les funcions que criden a cada part del Model o View corresponent.

Phoenix facilita l'ús de la tecnologia WebSockets, que permet enviar dades als clients sense que els clients hagin de fer peticions contínuament. Això vol dir que si, per exemple, processar una imatge triga un temps llarg (1 minut, per exemple), podem enviar al navegador una pàgina d'espera immediatament, i tan bon punt acabi el procés, enviar el resultat a l'usuari.

De cara al desenvolupament, una característica que inclou Phoenix per defecte i que fa augmentar molt la qualitat de vida és l'anomenat Live Reloading, o recarrega en calent. Això es compon de dues parts: Primer, si editem una pàgina que tenim oberta al navegador, aquesta es refrescarà automàticament tan bon punt desem, sense haver de recarregar el navegador. I segon, si modifiquem funcions dels nostres processos interns, que s'executen contínuament, com per exemple del servidor de processament d'imatges, no cal que els reiniciem. El codi es recompila automàticament tan bon punt desem, i els canvis s'apliquen al procés en execució.

7.5 Creació del prototip

Aquí detallem en què ha consistit el desenvolupament del prototip. En primer lloc, es van instal·lar en local, en un sistema Linux, Erlang, Elixir i PostgreSQL. Aquests mateixos passos es van haver de fer més tard també al servidor de producció.

En segon lloc es va crear un projecte de phoenix bàsic i un repositori git[1] que servirien com a base del projecte. A mesura que es va anar modificant el codi d'aquest projecte, es van afegir el formulari inicial, així com el codi receptor dels arxius d'imatge, i el procés virtual que actua de servidor d'imatges, preparat per a rebre la imatge del navegador, desar-la en memòria, i mantenir-se actiu per a poder tornar a l'usuari la seva imatge original i (més tard), la processada.

En tercer lloc, es va adaptar el primer mètode, una implementació de l'EAST[2]. Es per a cada mètode, el procediment va consistir a recollir les dependències en un entorn virtual de python (virtualenv), diferents per a cada mètode, modificar el codi per a poder acceptar una imatge en concret, i escriure un script *wrapper* per a cridar el mètode de manera senzilla. Per al cas de l'EAST, inicialment es va fer utilitzant stdin i stdout, per evitar que el servidor web hagués de crear fitxers temporals. En aquest moment encara no existia el sistema de RPC, i per tant no hi havia separació de xarxa entre els mètodes i el servidor web.

En quart lloc, de la mateixa manera que s'havia creat la pàgina inicial amb el formulari, es va crear la pàgina perquè l'usuari pogués veure els resultats, encara sense la valoració d'estrelles.

En cinquè lloc, es van crear els esquemes de la base de dades utilitzant la llibreria d'abstracció que ens proporciona Elixir, anomenada Ecto. Tenim taules per a les dades dels mètodes, el registre de les imatges processades, i les valoracions.

En sisè lloc es va dissenyar i implementar l'RPC, i es va adaptar l'EAST per a fer-lo servir. Un cop fet es van adaptar l'AdvancedEAST i el PixelLink tal com s'havia fet amb l'EAST, i es van integrar amb el backend mitjançant l'RPC de la mateixa manera.

En setè lloc es va millorar la interfície gràfica perquè mostrés animacions de càrrega mentre les imatges s'estaven processant, i es va utilitzar el sistema de websockets per a fer que l'usuari rebés les imatges immediatament havien estat processades, o un missatge d'error en cas que n'hi hagués.

En vuitè lloc, es va crear el sistema de valoracions amb estrelles, i la seva corresponent taula a la base de dades, tot seguint el procés de migracions d'Ecto.

Per últim, es va programar el sistema per a desar les imatges, es van afegir les caselles de consentiment, i es van escriure els Terms & Conditions.

7.6 Base de dades

El projecte utilitza una base de dades PostgreSQL a través de la llibreria Ecto d'Elixir, que abstreu tant les consultes com les migracions i la gestió de l'esquema.

A la figura 5 es mostra l'esquema de taules de la base de dades. Es desen a la base de dades:

- Els mètodes amb la informació per a ser executats.
- Les metadades de cada processat, amb les seves valoracions.
- En cas que l'usuari ho permeti, la imatge, per a ser utilitzada en entrenament futur.

8 PROVES I RESULTATS

8.1 Finalització del prototip

El prototip permet a l'usuari carregar una imatge des del seu navegador i veure'n el resultat un cop processat per tres algoritmes:

1. EAST (Efficient and Accurate Scene Text Detector [6]).
2. AdvancedEAST
3. PixelLink

A la pantalla principal també demanem a l'usuari que accepti els termes i condicions del servei, i si accepta que la seva imatge es desi i es faci servir en un futur per a millorar els algorismes.

Un cop processada la imatge, permetem a l'usuari valorar els resultats de cada algorisme, d'una a cinc estrelles. Aquest valor es desa a la base de dades.

[1] https://github.com/cosarara/computervision
[2] https://github.com/SakuraRiven/EAST

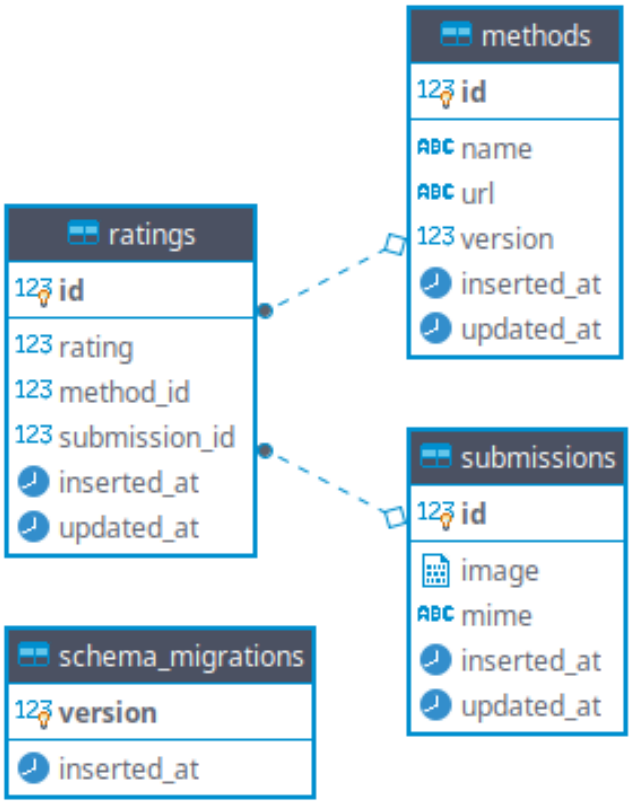

Fig. 5: Esquema de la base de dades

A les figures 6 i 7 podem veure unes captures del prototip.

El servidor és una màquina virtual del Centre de Visió per Computador, però no té un domini propi, de manera que de moment és accessible des d'un subdomini de l'autor[3] o des de l'adreça IP del servidor[4]. En cas que es volgués fer públic a més gran escala, com a projecte del CVC, s'hauria de cercar de donar-li un domini, potser sota el prefixe *.cvc.uab.es.

8.2 Rendiment

Amb els tres mètodes escollits es van fer proves per poder decidir els requisits d'execució.

Gràcies a l'eina `time` de GNU/Linux es pot veure, a més del temps d'execució, el màxim d'ús de memòria d'un procés.

La conclusió va ser que s'utilitzava al voltant de 2GB de RAM en el processament d'una imatge de 1000x1000 píxels, i tants threads com tingués l'ordinador. Per tant vam posar com a mínim perquè l'ordinador que executi els mètodes no exhaureixi mai la memòria 4GB de RAM. A la pràctica, un cop fetes proves amb usuaris reals, vam veure que aquest mínim era massa optimista, i la memòria de la màquina s'exhauria regularment. Es va augmentar la memòria a 8GB de RAM per evitar-ho.

Com a funcionalitat futura, es planteja que l'usuari vegi quants segons ha trigat a processar la imatge cada mètode, per a poder comparar-ne l'eficiència.

[3] `https://cv.cosarara.me`
[4] 158.109.8.58

8.3 Proves amb usuaris reals

A dia 14 de Juny de 2020, hem tingut 56 imatges enviades al servei per diferents usuaris, i 33 respostes a l'enquesta d'usabilitat que hem preparat a Google Forms. La cerca d'usuaris de prova es va fer a través de grups de whatsapp amb amics, familiars i amics d'amics.

Dels resultats de l'enquesta (figures 8 a 11), podem veure que la interfície podria ser més clara, ja que alguns usuaris no entenen de què tracta la plataforma. També comprovem que ha estat una bona idea assegurar que la interfície es vegi correctament en dispositius mòbils, ja que és el que han utilitzat la majoria d'usuaris. Hi va haver un gran nombre de fallades de processament, especialment amb el mètode PixelLink, pel fet que la màquina virtual en què s'executaven els processos es quedava sense memòria RAM. Això també es veu reflectit a l'enquesta.

Cal fer notar que una imatge que no s'ha pogut processar a causa d'un error, no rep valoració d'estrelles. És a dir, si de les tres imatges de resultats, una falla, només es permet a l'usuari valorar amb estrelles les altres dues.

Finalment, la majoria d'enquestats responen que tenen un cert interès pel projecte, tot i que la gran majoria no tornen a fer servir l'eina després de fer una o dues proves.

Un cop l'usuari veu els resultats de la detecció en la seva imatge, pot valorar els resultats d'1 a 5 estrelles, incloent-hi mitges estrelles. Per tant, una valoració possible serien 2 estrelles i mitja. Traduïm aquesta valoració a un nombre entre 1 (mitja estrella) i 10 (5 estrelles).

Dels ratings d'estrelles de la base de dades (figures 14 a 17 a l'apèndix A.3) , podem veure que l'EAST és el mètode que aconsegueix resultats perfectes més sovint, però que quan falla, ho fa més estrepitosament que no pas el PixelLink, que és més consistent. L'AdvancedEAST en canvi tendeix a donar resultats força dolents des del punt de vista dels nostres usuaris.

9 Conclusions

El servei web plantejat als objectius ha estat implementat, amb el seus front-end, back-end, i 3 mètodes. Tal com s'-havia descrit als objectius, els mètodes funcionen amb un sistema de Remote Procedure Call, de manera que és perfectament factible afegir mètodes de tercers en servidors externs. S'han fet proves amb usuaris reals de demografies variades: tant estudiants d'informàtica com adults de diferents entorns laborals, que demostren que l'eina és comprensible i usable. Amb tot això, podem dir que el projecte ha estat un èxit.

Si bé des del primer moment, l'objectiu del projecte no ha estat millorar o tirar endavant l'estat de l'art en detecció d'imatges de manera directa, l'existència d'aquesta nova plataforma dóna peu a anar més enllà. Aquest projecte podria servir doncs, com a la primera pedra per a muntar un sistema d'entrenament en què les valoracions humanes formessin part del feedback loop, el que es coneix com a HITL (Human-in-the-loop). És per això que permetem als usuaris decidir si les seves imatges poden ser desades per a entrenament.

Personalment, gràcies a aquest projecte he pogut experimentar amb tecnologies interessants: per una banda, els mètodes de reconeixement d'imatges i les seves particula-

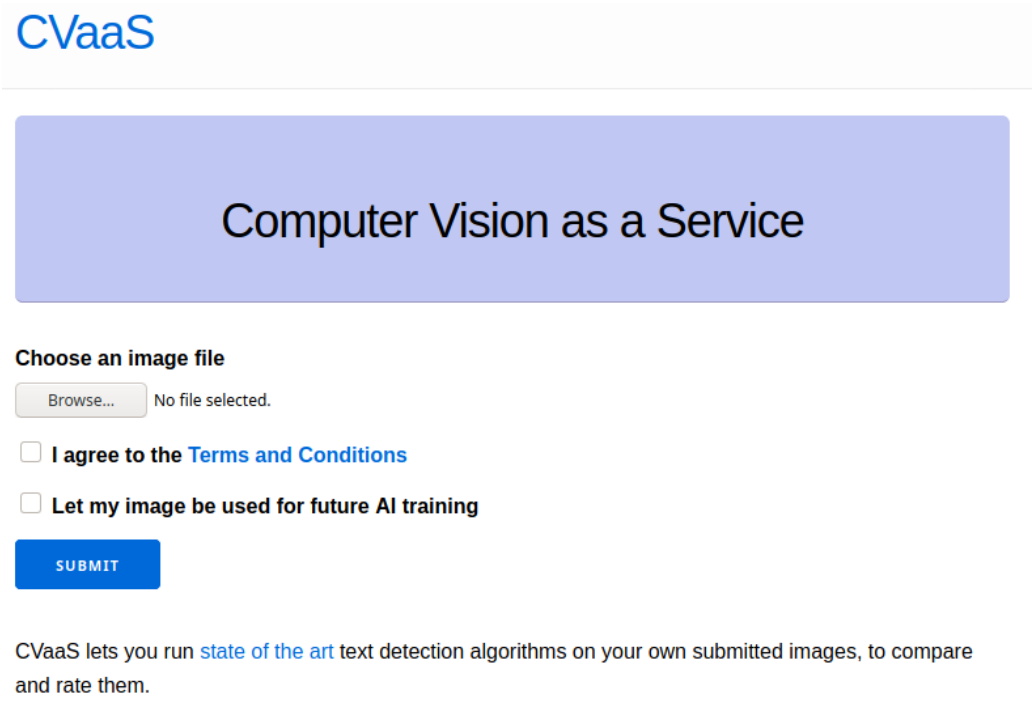

Fig. 6: Pantalla inicial

Fig. 7: Valoració d'estrelles

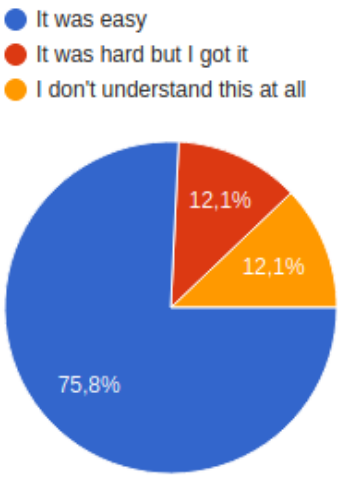

Fig. 8: Enquesta: Did you find the interface easy to understand?

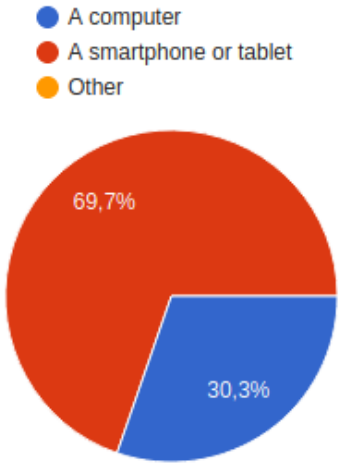

Fig. 9: Enquesta: What device did you use?

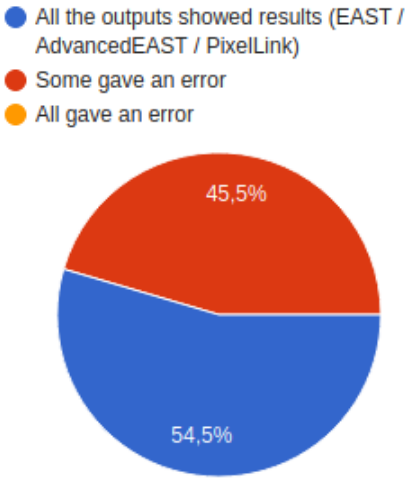

Fig. 10: Enquesta: Did it work?

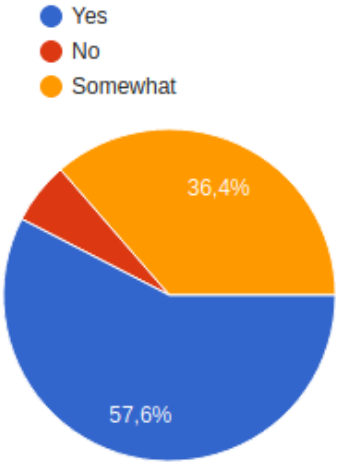

Fig. 11: Enquesta: Do you think the tool is interesting?

ritats a l'hora de ser adaptats per a un ús fora del context acadèmic en què es publiquen, i per l'altra, el sistema d'Elixir i el seu model d'actors.

Referències

[1] D. Karatzas, S. Robles, and L. Gomez, "An on-line platform for ground truthing and performance evaluation of text extraction systems," in *2014 11th IAPR International Workshop on Document Analysis Systems*, pp. 242–246, IEEE, 2014.

[2] L. von Ahn, "Human computation," in *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, ICDE '08, (USA), p. 1–2, IEEE Computer Society, 2008.

[3] L. v. Ahn, "Games with a purpose," *Computer*, vol. 39, p. 92–94, June 2006.

[4] J. Liu, X. Liu, J. Sheng, D. Liang, X. Li, and Q. Liu, "Pyramid mask text detector," 2019.

[5] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6154–6162, 2018.

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," 2017.

[7] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, "Hybrid task cascade for instance segmentation," 2019.

[8] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, "East: An efficient and accurate scene text detector," 2017.

[9] X. Li, W. Wang, W. Hou, R.-Z. Liu, T. Lu, and J. Yang, "Shape robust text detection with progressive scale expansion network," 2018.

[10] S. Long, J. Ruan, W. Zhang, X. He, W. Wu, and C. Yao, "Textsnake: A flexible representation for detecting text of arbitrary shapes," 2018.

[11] W. Wang, E. Xie, X. Li, W. Hou, T. Lu, G. Yu, and S. Shao, "Shape robust text detection with progressive scale expansion network," 2019.

[12] P. Lyu, C. Yao, W. Wu, S. Yan, and X. Bai, "Multi-oriented scene text detection via corner localization and region segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[13] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, "East: An efficient and accurate scene text detector," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[14] X. Zhu, H. Hu, S. Lin, and J. Dai, "Deformable convnets v2: More deformable, better results," 2018.

[15] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao, "Detecting text in natural image with connectionist text proposal network," 2016.

[16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," *Lecture Notes in Computer Science*, p. 21–37, 2016.

[17] J. Baek, G. Kim, J. Lee, S. Park, D. Han, S. Yun, S. J. Oh, and H. Lee, "What is wrong with scene text recognition model comparisons? dataset and model analysis," 2019.

APÈNDIX

A.1 Diagrames de Gantt

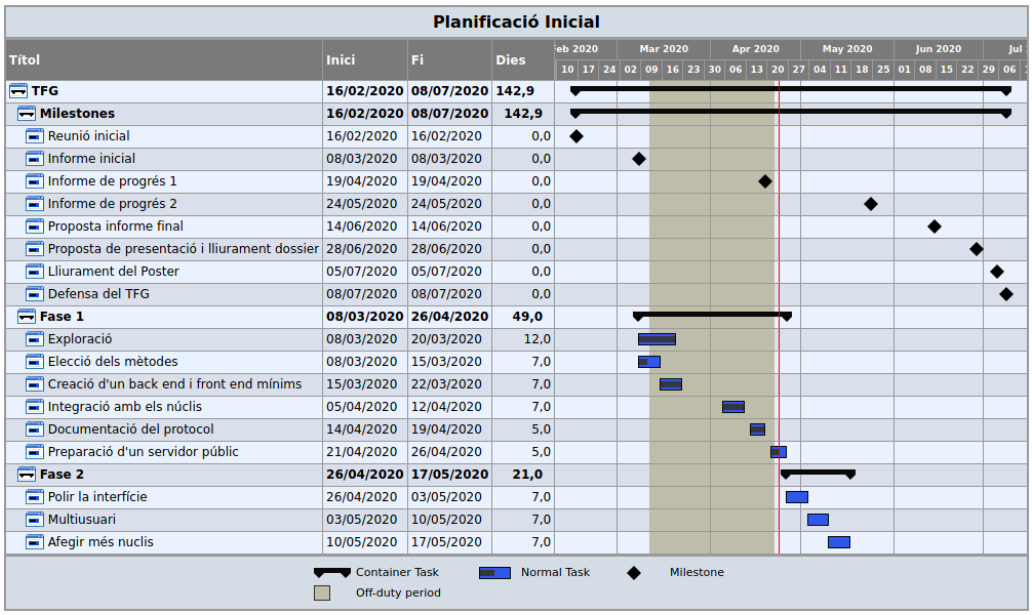

Fig. 12: Planificació inicial

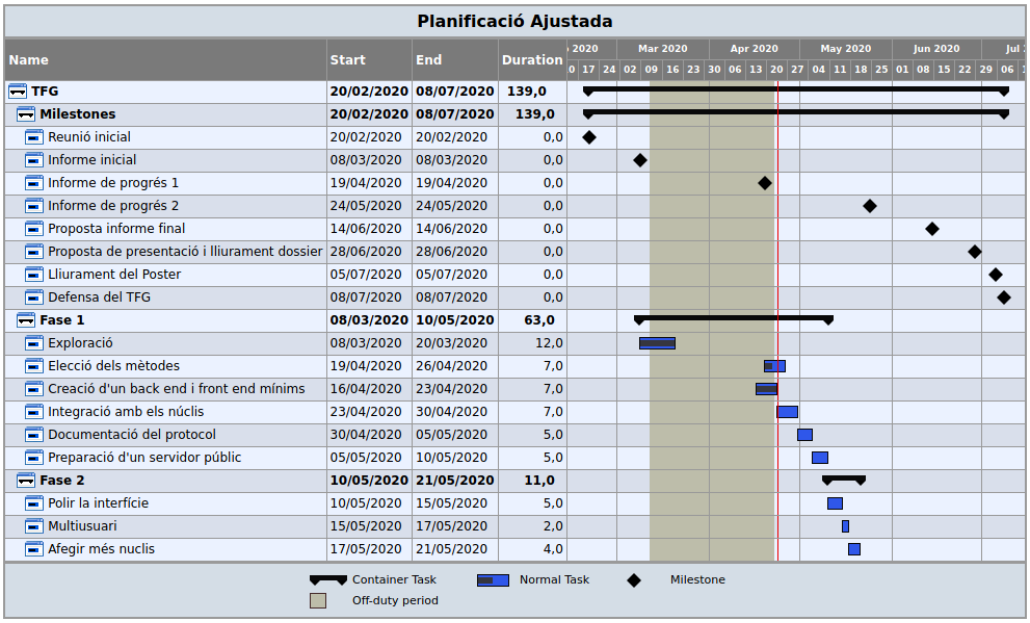

Fig. 13: Planificació ajustada

A.2 Mostra de l'RPC

```
from flask import Flask, request, jsonify
import subprocess
import base64
app = Flask(__name__)

@app.route('/')
def index():
    return jsonify(name="EAST")
```

```
@app.route('/process', methods=['POST'])
def process():
    data = request.get_json(force=True)
    script = "/home/jaume/Sync/proj/tfg/cores/EAST/wrapper.sh"
    imagedata = base64.b64decode(data['image'])
    name = data['filename']
    print(f"processing {name}")
    o = subprocess.run([script], capture_output=True, input=imagedata)
    print(f"done processing {name}!")
    outimage = base64.b64encode(o.stdout).decode("ascii")
    return jsonify(image=outimage, mime="image/png")
```

A.3 Histogrames

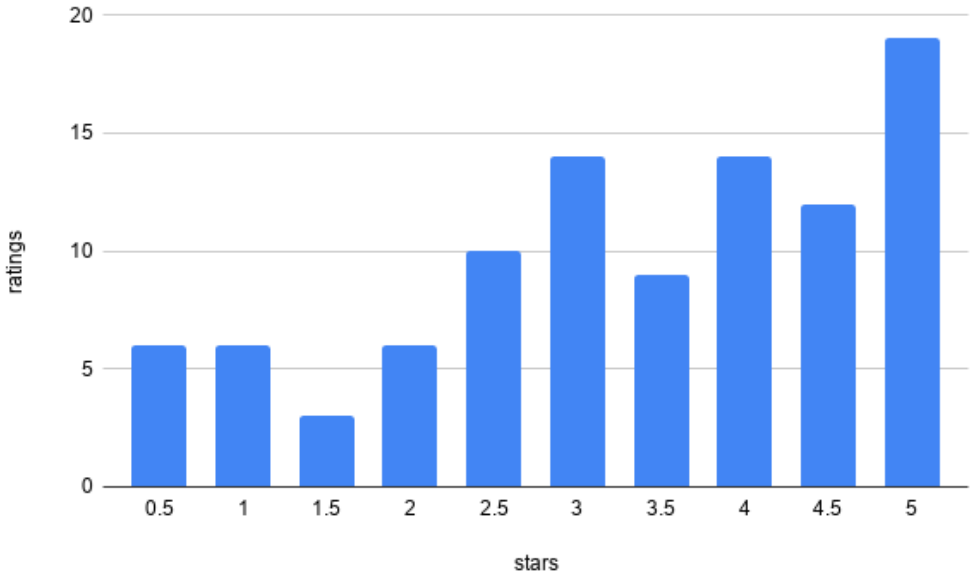

Fig. 14: Histograma de valoracions de tots els mètodes

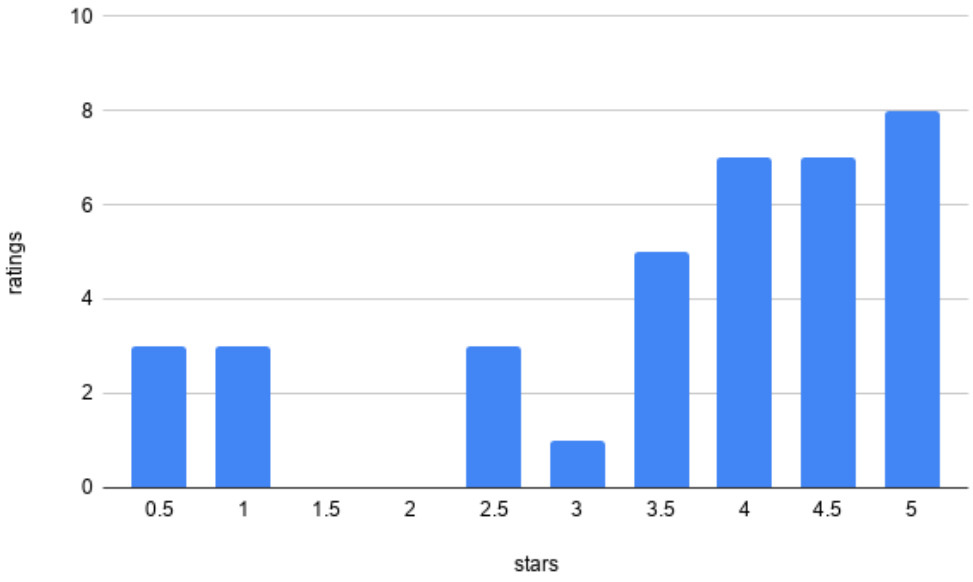

Fig. 15: Histograma de valoracions de l'EAST

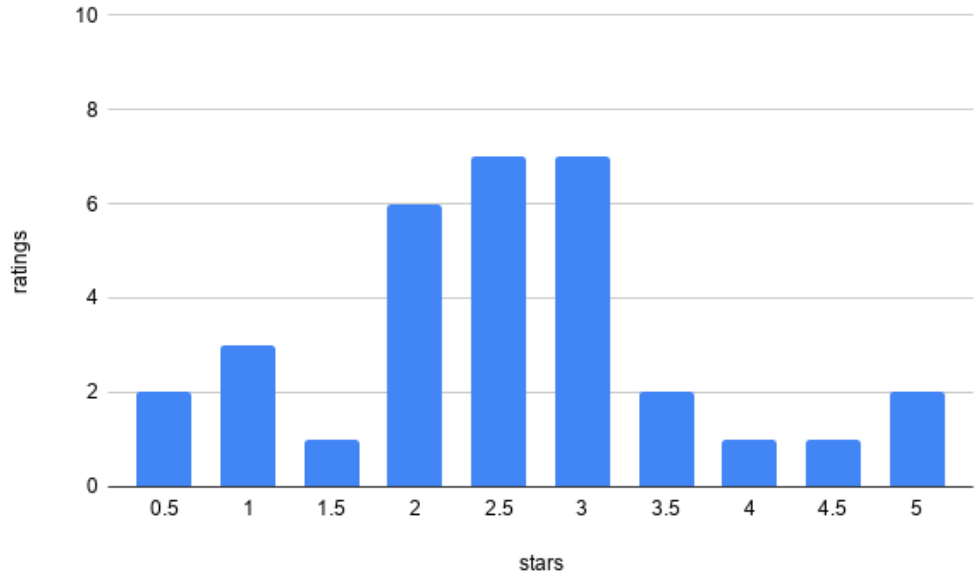

Fig. 16: Histograma de valoracions de l'AdvancedEAST

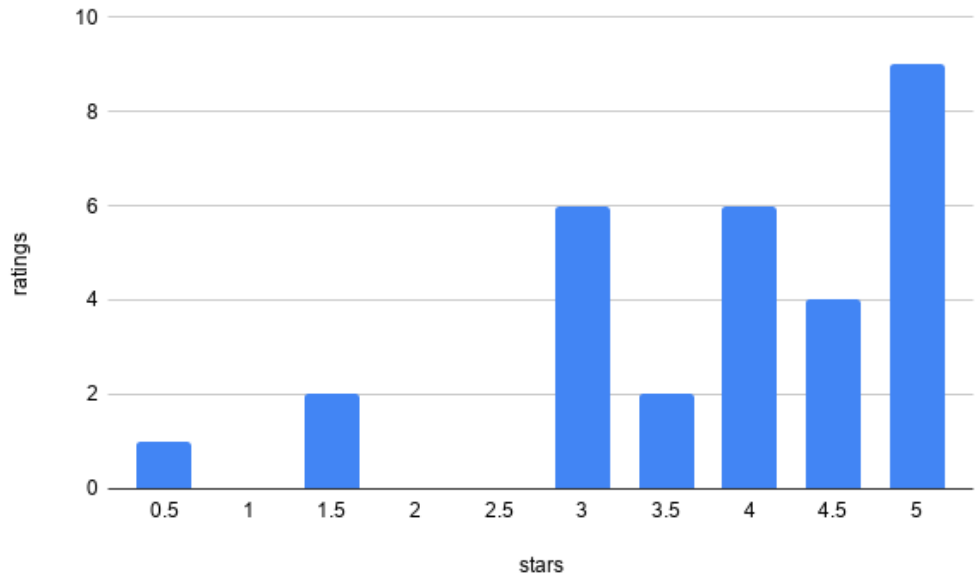

Fig. 17: Histograma de valoracions del PixelLink