— Grau
en **Traducció**
i Interpretació

Treball de Fi de Grau

Implementació d'un model híbrid per al reconeixement d'entitats nomenades en llengua catalana

Albert Jacas Mateu

Tutor/a: Ángel Souto Cabaleiro
Seminari 207: Traducció i tecnologies

Curs 2023-2024

upf. **Universitat Pompeu Fabra** *Barcelona*

Facultat de Traducció i Ciències del Llenguatge

Resum

En l'era digital actual, el processament del llenguatge natural (PLN) s'enfronta al repte de processar un volum i diversitat de textos sense precedents, circumstància que planteja obstacles significatius per a l'extracció i l'organització de la informació. El reconeixement d'entitats nomenades (REN) és crucial en aquest context, ja que identifica i classifica automàticament noms propis, ubicacions, dates, quantitats i altres expressions específiques. En aquest treball de fi de grau es desenvolupa un model de reconeixement d'entitats nomenades per a la llengua catalana mitjançant l'ampliació de les capacitats dels models existents, que sovint són limitats per a aquest idioma. La metodologia combina tècniques d'aprenentatge automàtic i regles per crear un model híbrid capaç de detectar entitats de nom propi (ENAMEX), expressions temporals (TIMEX) i expressions numèriques (NUMEX). L'avaluació del model amb el corpus AnCora, el qual s'ha anotat manualment, revela una mesura F del 82,03%, fet que destaca la viabilitat i efectivitat del sistema. Finalment, es presenta una aplicació pràctica del model desenvolupat mitjançant la creació de glossaris *ad hoc* per a eines de traducció assistida per ordinador (TAO).

Paraules clau — Reconeixement d'entitats nomenades, processament del llenguatge natural, traducció assistida per ordinador, llengua catalana

Resumen

En la era digital actual, el procesamiento del lenguaje natural (PLN) se enfrenta al reto de procesar un volumen y diversidad de textos sin precedentes, circunstancia que plantea obstáculos significativos para la extracción y organización de la información. El reconocimiento de entidades nombradas (REN) es crucial en este contexto, ya que identifica y clasifica automáticamente nombres propios, ubicaciones, fechas, cantidades y otras expresiones específicas. En este trabajo de fin de grado se desarrolla un modelo de reconocimiento de entidades nombradas para la lengua catalana mediante la ampliación de las capacidades de los modelos existentes, que a menudo son limitados para este idioma. La metodología combina técnicas de aprendizaje automático y reglas para crear un modelo híbrido capaz de detectar entidades de nombre propio (ENAMEX), expresiones temporales (TIMEX) y expresiones numéricas (NUMEX). La evaluación del modelo con el corpus AnCora, que se ha anotado manualmente, revela una medida F del 82,03%, lo que destaca la viabilidad y efectividad del sistema. Por último, se presenta una aplicación práctica del modelo desarrollado mediante la creación de glosarios *ad hoc* para herramientas de traducción asistida por ordenador (TAO).

Palabras clave — Reconocimiento de entidades nombradas, procesamiento del lenguaje natural, traducción asistida por ordenador, lengua catalana

Abstract

In today's digital age, natural language processing (NLP) faces the challenge of processing unprecedented volumes and diversity of texts, which poses significant difficulties for information extraction and organization. Named entity recognition (NER) is crucial in this context, as it involves identifying and classifying specific text segments automatically, such as proper names, locations, dates, and quantities. This dissertation aims to develop a named entity recognition model for the Catalan language by enhancing the capabilities of existing models, which are often limited for this language. The methodology combines machine learning techniques and rule-based approaches to create a hybrid model capable of detecting named entities (ENAMEX), temporal expressions (TIMEX), and numerical expressions (NUMEX). The model's evaluation, using the manually annotated AnCora corpus, demonstrates an F-measure of 82.03%, underscoring its viability and effectiveness. Furthermore, a practical application of the developed NER model is presented through the creation of ad hoc termbases for computer-assisted translation (CAT) tools.

Keywords — Named entity recognition, natural language processing, computer-assisted translation, Catalan language

Agraïments

En primer lloc, vull expressar tot el meu agraïment al meu tutor, Ángel Souto, el qual ha vetllat durant tot el procés d'elaboració d'aquest treball per aconseguir el millor resultat possible.

També vull agrair al Borja tota l'ajuda que m'ha prestat i per fer-me costat durant tot el procés. D'una banda, el codi ha resultat el que és gràcies als seus consells i a la seva expertesa en el camp de la programació. De l'altra, l'acompliment d'aquest projecte es deu al seu suport i afecte inesgotables.

Igualment, vull expressar el meu agraïment més sentit a tots els meus amics que, tot i la distància o la no-distància que ens separa, sempre són els primers a mostrar tot el seu suport.

En últim lloc, vull donar les gràcies als meus pares i germans per tot el suport i afany incondicionals.

Índex

Índex de figures

Índex de taules

Capítol 1

Introducció

En l'era digital actual, la lingüística i la computació han de processar un volum i una diversitat de textos sense precedents. Aquesta sobreabundància d'informació planteja desafiaments significatius per a l'extracció i l'organització de la informació. Dins d'aquest context, el reconeixement d'entitats nomenades sorgeix com una disciplina crucial en el camp del processament del llenguatge natural, la qual consisteix a identificar i classificar segments de text que es refereixen a noms propis o a fragments de text específics, com ara organitzacions, ubicacions, expressions de temps, quantitats o valors monetaris, entre altres. La capacitat de reconèixer aquestes entitats és fonamental per a aplicacions com la traducció automàtica, el text predictiu o la intel·ligència artificial degut a l'atenció personalitzada que requereix aquest tipus d'informació.

Malgrat que moltes aplicacions que requereixen el reconeixement d'entitats nomenades ja disposen de sistemes interns capaços de detectar-les, molts dels models de programari lliure existents només poden detectar-ne una quantitat reduïda. Això dificulta el desenvolupament d'altres aplicacions que depenen del reconeixement d'entitats nomenades.

Així, l'objectiu d'aquest treball de fi de grau és el desenvolupament d'un model de reconeixement d'entitats nomenades per a la llengua catalana mitjançant l'ampliació de les capacitats dels models existents. Per aconseguir-ho, es crearà un model híbrid, el qual combinarà les tècniques de l'aprenentatge automàtic i la cerca basada en regles.

A nivell personal, el desenvolupament d'eines digitals per a la llengua catalana constitueix una de les principals motivacions d'aquest treball. Tradicionalment, la majoria d'aquestes eines han gaudit d'una àmplia presència en altres llengües com l'anglès o el castellà, però rarament en català. Així doncs, la contribució a la presència del català en aquesta mena d'eines es presenta com una raó significativa per a l'acompliment d'aquest treball de fi de grau.

Pel que fa a l'estructura del treball, la recerca es divideix en diversos capítols, el primer dels quals és aquesta introducció. El segon capítol presenta el marc teòric per entendre el context actual del processament del llenguatge natural i del reconeixement d'entitats nomenades. Aquest marc teòric és fonamental per prendre les millors decisions i aconseguir un bon resultat final.

El tercer capítol presenta la metodologia seguida per tal d'obtenir el producte final. Tot seguit, el quart capítol presenta les característiques del model implementat, capaç de trobar entitats nomenades en un text d'entrada. El cinquè capítol analitza els resultats obtinguts tot seguint un mètode d'avaluació rigorós, basat en metodologies àmpliament utilitzades en aplicacions d'ús quotidià.

Finalment, el sisè capítol analitza un possible ús real per a les entitats nomenades trobades mitjançant la creació d'un glossari *ad hoc* per a un text del qual s'han extret aquestes entitats. Aquest glossari podrà ser utilitzat en una eina de traducció assistida per complir amb el principi de coherència que es pretén assolir gràcies a l'ús d'aquesta mena de programari.

Capítol 2

Marc teòric

La lingüística computacional, com qualsevol àmbit relacionat amb la computació, es troba en canvi constant. Aquest capítol presenta els orígens i l'evolució del reconeixement d'entitats nomenades per obtenir una visió general de l'estat de la qüestió. Aquesta informació representa el coneixement necessari per prendre una decisió informada que permetrà escollir els paràmetres més convenients per a la implementació del model de reconeixement d'entitats nomenades per al català.

2.1 El processament del llenguatge natural

El processament del llenguatge natural (PLN), també conegut per les seves sigles angleses NLP (Natural Language Processing), és un camp de la informàtica que estudia la interacció entre els ordinadors i el llenguatge natural. El PLN té l'objectiu d'ensenyar els ordinadors a comprendre i processar el llenguatge de forma similar a com ho fa un ésser humà.

El PLN va aparèixer a finals de la dècada dels 40 amb la intenció de desenvolupar un sistema de traducció automàtica (TA). Malgrat un període inicial d'entusiasme i optimisme i un primer sistema rudimentari de traducció automàtica del rus a l'anglès, aquest projecte no va tenir l'èxit esperat degut a les limitacions computacionals de l'època (Jones, 1994). Amb l'informe ALPAC de 1966 (*Language and Machines*, 1966), el desenvolupament de la TA va estancar-se i el PLN va centrar-se en altres camps de coneixement.

Amb el temps, van aparèixer nombrosos sistemes de PLN, cadascun centrat en la realització d'una tasca concreta. Tres d'aquestes tasques són d'especial rellevància per aquest estudi:

- Segmentació de paraules (o tokenització). Es tracta del punt de partida de moltes altres tasques

que s'efectuen mitjançant el PLN. Consisteix en dividir un text en *tokens*, una unitat més fàcilment interpretable. La gran majoria d'aquests tokens corresponen a una paraula, tot i que també poden incloure signes de puntuació, clítics, nombres i símbols (Grefenstette, 1999).

- Desambiguació lèxica. Consisteix en l'assignació d'una categoria gramatical a cadascuna de les paraules que conformen un text informatitzat. Partint d'un text ja segmentat en tokens, aquesta tasca és molt més factible. Al seu torn, la desambiguació lèxica també facilita molt l'execució d'altres tasques de PLN més complexes (Martinez, 2012).

- Reconeixement d'entitats nomenades. És la tasca al voltant de la qual gira aquest estudi. A causa de la importància que té, el següent apartat es dedicarà completament a la descripció d'aquesta tasca.

2.2 Reconeixement d'entitats nomenades

El reconeixement d'entitats nomenades (REN) és una tasca dedicada a l'extracció i classificació d'informació en textos informatitzats. És capaç de reconèixer noms de persona, organitzacions, dates, llocs, magnituds i quantitats monetàries i etiquetar-los adequadament. Un model de REN operatiu rep un conjunt de n-tuples ($[t_1]$, $[t_2]$, ..., $[t_N]$) i tot seguit produeix un conjunt d'etiquetes (E_1, E_2, ..., E_N) per a les seves tuples corresponents (Jehangir, Radhakrishnan i Agarwal, 2023). La Figura 2.1 mostra un cas en què un model de REN identifica correctament dues entitats nomenades en una frase.

El REN, com a tasca amb nom propi, es va presentar a la sisena edició del Message Understanding Conference ("Appendix C", 1995) i va suposar un canvi de paradigma en el camp de l'extracció de la informació (EI), el qual es dedica a la recuperació d'informació en textos informatitzats. Des de llavors, ha evolucionat constanment fins obtenir els sistemes d'avui en dia.

Abans d'aquesta conferència, la recerca sobre aquest tema era escassa i molt centrada en àmbits concrets. Destaquen, principalment, els articles dedicats al reconeixement de noms d'empreses, nascuts amb la intenció de millorar les anàlisis automàtiques de text i la generació automàtica de bases de dades (Rau, 1991).

A conseqüència de la conferència, la recerca en REN va guanyar protagonisme. Sobretot al principi, quan la creació de models de REN basats en regles era el mètode utilitzat per excel·lència, la importància dels lingüistes va ser cabdal. Amb l'aparició de l'aprenentatge automàtic i l'ús de textos per entrenar aquests models, la importància del seu paper va disminuir, però en cap cas va desaparèixer.

En l'actualitat, el REN es considera una part integral de moltes aplicacions pràctiques, com ara els sistemes de resposta automàtica, els assistents personals intel·ligents o les aplicacions de seguretat informàtica.

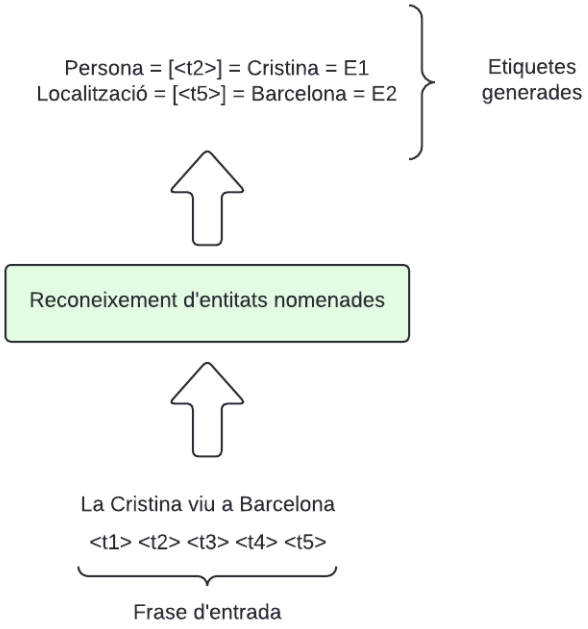

Figura 2.1: Exemple de funcionament del REN.

Per exemple, en el sector financer, el REN pot ajudar a detectar informació sensible com números de compte o dades personals en grans volums de text per garantir el compliment de les normatives de protecció de dades (Li, Sun, Han i Li, 2022).

2.2.1 Models de REN

Al llarg del temps, i de forma comparable als sistemes de traducció automàtica, el reconeixement d'entitats nomenades va anar adaptant-se a les tecnologies que sorgien. Avui, els models de REN es poden classificar en tres tipus: models basats en regles, models d'aprenentatge automàtic i models híbrids. Cadascun és útil per a certes situacions, tot i que els models d'aprenentatge automàtic són els més utilitzats avui dia.

Models basats en regles

Els models de REN basats en regles extreuen entitats basant-se completament en la creació de regles dictades per lingüistes. Aquestes regles recuperen les entitats mitjançant característiques gramaticals (per exemple, a partir de la categoria gramatical de les paraules), sintàctiques (per exemple, a partir de l'ordre de les paraules) i ortogràfiques (per exemple, a partir de paraules que comencen per majúscula). Combinades amb diccionaris informatitzats, les regles són capaces de distingir entre lèxic comú i entitats nomenades (Mansouri, Affendey i Mamat, 2008).

Un dels primers models de REN basat en regles va ser el de FASTUS (Appelt, Hobbs, Bear, Israel, Kameyama, Kehler et al., 1995). Es va presentar a la conferència MUC-6 i funcionava íntegrament a partir de regles creades mitjançant expressions regulars. Van dividir el procés en tres passos: reconeixement de frases, reconeixement de patrons i reconeixement d'entitats amagades dins del text. Aquest últim pas se centrava en paraules que quedaven lligades amb altres mots i que podien ser considerades entitats nomenades. Un exemple d'aquest fet en anglès és el possessiu (*'s*) lligat al nom propi.

Els models basats en regles generen més bons resultats en camps específics. Fins i tot, si s'han creat amb especificitat suficient, són capaços d'obtenir més bons resultats que els d'aprenentatge automàtic (Chiticariu, Krishnamurthy, Li, Reiss i Vaithyanathan, 2010). Tot i això, no són els models més utilitzats avui en dia a causa del cost de mà d'obra i de temps que comporten.

Com a aspecte positiu, convé destacar que, per a llengües pròximes, és més senzill adaptar-ne les regles. A més, no cal reunir un corpus d'entrenament per construir el model final.

Models basats en aprenentatge automàtic

Els models de reconeixement d'entitats nomenades basats en aprenentatge automàtic utilitzen algoritmes d'aprenentatge supervisats i no supervisats per identificar i classificar entitats dins dels textos. Aquests models s'entrenen amb grans conjunts de dades anotades, en què les entitats han estat prèviament marcades per humans. A diferència dels models basats en regles, els d'aprenentatge automàtic poden adaptar-se millor a la variabilitat i l'ambigüitat del llenguatge natural.

Els models d'aprenentatge automàtic més populars per al REN inclouen les xarxes neuronals recurrents (RNN), les quals han demostrat que són molt efectives per a la seqüenciació de textos. Recentment, alguns models com BERT (Bidirectional Encoder Representations from Transformers) o GPT (Generative Pre-trained Transformer) han establert nous estàndards de rendiment per a aquesta tasca gràcies a la capacitat que tenen de comprendre millor el context dels textos analitzats (Devlin, Chang, Lee i Toutanova, 2019).

El principal avantatge dels models basats en aprenentatge automàtic és la flexibilitat i capacitat d'adaptació. No requereixen una programació de regles específiques, fet que permet reduir el manteniment necessari. No obstant això, la qualitat del model depèn enormement de la quantitat i qualitat de les dades d'entrenament disponibles, així com de la capacitat de l'algoritme per generalitzar a partir d'aquests exemples.

Models híbrids

Els models híbrids combinen les tècniques de models basats en regles i models d'aprenentatge automàtic per aprofitar els punts forts de cadascun. Aquests models són especialment útils en situacions en què es disposa de dades anotades limitades per entrenar un model completament basat en aprenentatge automàtic o quan es treballa amb idiomes per als quals no existeixen recursos suficients.

Un exemple comú d'una aproximació híbrida és l'ús de regles per preprocessar o postprocessar les sortides del model d'aprenentatge automàtic. Això pot incloure l'ús de regles per ajustar la confiança en les prediccions del model o per incorporar coneixement específic del domini que pot ser difícil d'aprendre només amb exemples de text. Un altre enfocament és utilitzar un model d'aprenentatge automàtic en una primera passada per identificar candidats a entitats i després refinar aquests resultats amb un conjunt de regles específiques del domini (Mansouri et al., 2008).

Els models híbrids són una solució robusta que pot beneficiar-se de l'eficiència de l'aprenentatge automàtic mentre es reté el control més precís que proporcionen les regles. D'aquesta forma, esdevé una opció viable en moltes aplicacions pràctiques de REN. Cal destacar que la llibreria de Python spaCy, especialitzada en tasques del processament del llenguatge natural, inclou un model d'aprenentatge automàtic disponible en llengua catalana. A dia d'avui, però, aquest model no inclou moltes de les etiquetes disponibles en altres idiomes, particularment l'anglès. Aquesta situació presenta una possibilitat de complementar el model en català mitjançant un sistema híbrid.

2.2.2 Entitats reconeixibles

Dins del camp del REN es distingeixen diversos tipus d'entitats que poden ser identificades i classificades pels diferents models. Aquestes inclouen entitats de nom propi (ENAMEX), expressions temporals (TIMEX) i expressions numèriques (NUMEX) (Chinchor i Robinson, 1998). La capacitat de reconèixer aquestes entitats no només ajuda a comprendre millor el text, sinó que també facilita moltes altres tasques com, per exemple, la traducció automàtica.

Entitats de nom propi (ENAMEX)

Les entitats de nom propi, o ENAMEX, són un tipus fonamental d'entitats en el reconeixement d'entitats nomenades. Les ENAMEX inclouen noms de persones, organitzacions, llocs geogràfics i altres entitats que són identificables com a noms propis únics. La identificació i classificació correcta d'aquestes entitats és crucial per a nombroses aplicacions de processament de llenguatge natural, com la cerca d'informació, l'extracció de dades o el resum automatitzat de textos, entre d'altres.

Les entitats ENAMEX es poden subdividir en diverses categories, que inclouen:

- Persones. Noms de persones, títols o referències que identifiquen a individus.
- Organitzacions. Noms d'empreses, governs, institucions i altres grups corporatius.
- Localitzacions. Noms de llocs geogràfics com ciutats, països, rius, muntanyes, etc.

El tractament d'aquestes entitats en textos sovint requereix un gran coneixement del context, ja que el mateix nom pot representar categories diferents. Per exemple, *Montserrat* podria referir-se a una persona, al massís de Catalunya o a l'illa del Carib, entre d'altres.

Per identificar i classificar ENAMEX, els models de REN generalment utilitzen tècniques d'aprenentatge automàtic. Aquests models s'entrenen en corpus anotats que mostren exemples de text amb entitats anotades i permeten que aprenguin com es representen típicament en el llenguatge aquestes entitats.

A més de l'aprenentatge automàtic, els models basats en regles encara juguen un rol important, especialment en dominis específics en què els patrons de noms poden ser molt regulars i previsibles. Aquests sistemes poden complementar els models d'aprenentatge automàtic oferint una capa addicional de validació o correcció basada en regles lògiques i coneixement específic del domini.

La capacitat per processar i entendre correctament les entitats ENAMEX en textos és un dels pilars bàsics del PLN modern. Permeten una millor interacció entre humans i màquines i faciliten una interpretació més rica i profunda dels textos.

Expressions temporals (TIMEX)

Les expressions temporals, o TIMEX, són un altre tipus d'entitat reconeguda dins del camp del reconeixement d'entitats nomenades. El reconeixement i la normalització de les expressions temporals són essencials per a moltes aplicacions com la gestió de calendaris, l'organització d'arxius històrics, la interpretació de notícies o la planificació d'esdeveniments.

Existeixen diversos tipus de TIMEX, cadascun dels quals presenta reptes únics en termes de reconeixement i interpretació:

- Dates senceres i hores. Cal identificar i normalitzar formats diversos, com *05/04/2024*, *5 d'abril de 2024*, *08.30* o *7:35*.
- Moments i dates relatius al dia actual. Existeix molta variabilitat per expressar una data o moment relatiu al dia actual, com *divendres que ve*, *abans-d'ahir* o *a primera hora del matí*

- Freqüències. Inclouen patrons repetitius, com *diàriament*, *cada any* o *cada dilluns*.

Per a la detecció i el processament de TIMEX, els models de REN generalment utilitzen etiquetadors especialitzats que inclouen informació temporal. Aquests models poden ser tant basats en regles com d'aprenentatge automàtic. Per exemple, eines basades en regles com HeidelTime han estat dissenyades específicament per a la identificació i normalització d'expressions temporals en textos i són capaces de treballar en molts idiomes, fins i tot sense haver creat regles específiques per a cadascun (Strötgen i Gertz, 2010).

El correcte tractament de les expressions temporals no només millora la usabilitat de les dades recuperades sinó que també enriqueix l'anàlisi contextual dels textos i faciliten la comprensió i automatització de tasques que depenen del temps.

Expressions numèriques (NUMEX)

Les expressions numèriques, o NUMEX, són un tipus d'entitat nomenada que inclou qualsevol dígit o conjunt de dígits que representen quantitats numèriques. Inclouen des de nombres purs fins a quantitats monetàries, percentatges, mesures, dates en format numèric, números de telèfon i altres identificadors com números d'identificació personal o números de sèrie. El reconeixement d'expressions numèriques és crucial per a molts processos de dades, ja que permeten l'extracció d'informació quantitativa essencial per a anàlisis estadístiques, informes financers, o qualsevol tasca que requereixi manipulació numèrica.

Els models de REN que tracten amb NUMEX han de ser capaços de distingir entre diferents formats i usos de números. Per exemple, *2024* podria ser interpretat com un any en un context, però podria ser simplement un nombre en un altre. Això fa que la contextualització sigui particularment important en el processament de NUMEX.

Els models d'aprenentatge automàtic, com BERT, són útils per reconèixer aquestes entitats gràcies a la seva capacitat per entendre el context en el qual apareixen els números. A més, la incorporació de regles específiques per a un camp en concret pot millorar significativament la precisió de la detecció d'aquestes expressions, com per exemple distingir entre una data i una quantitat simplement per l'estructura o els símbols adjacents com la barra (/) present en algunes en dates i la coma (,) dels decimals.

Així, el tractament de NUMEX en textos no només ajuda a identificar i extreure dades sinó que també permet una comprensió més profunda de la informació continguda en documents, la qual cosa els fa més accessibles per a les màquines.

2.3 REN en diverses llengües

El reconeixement d'entitats nomenades varia significativament entre llengües, principalment a causa de diferències en estructura gramatical, ús de la llengua i disponibilitat de recursos com corpus anotats o eines específiques de processament de llenguatge. A continuació, es detallen els enfocaments i reptes específics del REN per a l'anglès, l'espanyol i el català.

2.3.1 REN per a l'anglès

L'anglès ha estat tradicionalment la llengua amb més recursos en el camp del reconeixement d'entitats nomenades. Això es deu a la gran quantitat de dades disponibles, tant en termes de corpus de text marcat com de tecnologia de processament de llenguatge natural desenvolupada específicament per a aquesta llengua.

En anglès, els models de REN han evolucionat des d'enfocaments simples basats en regles fins a models complexos d'aprenentatge profund com BERT i GPT, que utilitzen entrenament extensiu per millorar la seva capacitat de generalització. Aquests models han establert el marc per a aplicacions avançades que inclouen la identificació precisa de noms, localitzacions, dates i altres entitats en contextos variats.

Pel que fa al programari lliure, la llibreria de Python spaCy inclou un model d'aprenentatge automàtic capaç de detectar tota mena d'entitats nomenades. A més, spaCy es beneficia d'una comunitat activa de desenvolupadors per a la llengua anglesa que contribueixen constantment a la millora dels models, fet que permet que es mantingui al dia amb els darrers avenços en el camp del processament de llenguatge natural.

2.3.2 REN per a l'espanyol

El reconeixement d'entitats nomenades en espanyol presenta reptes únics, com la flexió en la formació del gènere i el nombre, així com l'ús extensiu d'oracions relatives que poden complicar la identificació d'entitats. Aquestes particularitats lingüístiques, també presents en el català, requereixen models adaptats específicament per a manejar les variacions morfològiques i sintàctiques que no es troben en altres llengües com l'anglès.

Tot i que l'espanyol té menys recursos en comparació amb l'anglès, durant els últims anys s'ha vist un augment en la disponibilitat de corpus anotats i eines específiques per al PLN en aquesta llengua. Aquestes millores han estat impulsades per una creixent comunitat de recerca i desenvolupament que treballa en la creació de dades i recursos adaptats per a l'espanyol. Per exemple, projectes com AnCora

(Taule, Marti i Recasens, 2008) i CoNLL-2002 (Tjong Kim Sang, 2002) han proporcionat corpus anotats que han estat fonamentals per a l'entrenament i l'avaluació de models de REN.

Alguns models d'aprenentatge automàtic s'han adaptat a l'espanyol gràcies a tècniques avançades com l'aprenentatge per transferència, en què els models entrenats en anglès són afinats amb dades en espanyol. Aquest enfocament ha millorat significativament la capacitat de reconeixement d'entitats en aquesta llengua, ja que aprofita els avenços tecnològics desenvolupats inicialment per a l'anglès i els adapta a les particularitats de l'espanyol.

D'altra banda, hi ha una creixent disponibilitat de programari de codi obert que facilita el desenvolupament de sistemes de REN per a l'espanyol. Llibreries com spaCy i Hugging Face Transformers ofereixen suport per a l'espanyol proporcionant models preentrenats i eines per a la personalització i el refinament de models. Aquestes llibreries s'han utilitzat àmpliament tant en entorns acadèmics com de producció i han contribuït a la millora dels models de REN en espanyol.

2.3.3 REN per al català

El català, com a llengua menys representada en la investigació i desenvolupament de PLN, enfronta desafiaments addicionals en el camp del REN. La falta de grans corpus anotats i eines específiques limita les opcions disponibles per a la investigació i l'aplicació pràctica. No obstant això, hi ha hagut esforços significatius per desenvolupar recursos específics per al català. Projectes com el CLiC (Centre de Llenguatge i Computació) de la Universitat de Barcelona o el corpus anotat AnCora han contribuït a la creació de noves eines.

Amb tot, la recerca específica de REN per a la llengua catalana encara és limitada. Un dels pocs articles científics que exploren el tema va ser escrit el 2003 i se centrava en el trasllat d'un sistema d'aprenentatge automàtic adaptat des de l'espanyol (Carreras, Màrquez i Padró, 2003). Tot i que els resultats eren prometedors, els mètodes utilitzats en aquesta investigació s'escapen de l'àmbit temàtic d'aquest treball acadèmic.

Tot i això, val la pena comentar les estratègies que s'han seguit en altres llengües amb menys recursos per desenvolupar un model de REN basat en regles. Per exemple, per al malai, es va decidir crear un model basat en regles a causa de la falta de corpus anotats en aquesta llengua. Per aconseguir-ho, van recolzar-se en les regles ja creades d'altres idiomes similars com l'indonesi (Alfred, Leong, On i Anthony, 2014). Altres exemples d'aquest procés inclouen l'urdú (Riaz, 2010), que aprofitava les regles creades per a l'hindi, o el bengalí (Drovo, Chowdhury, Uday i Das, 2019).

2.4 Sistemes d'avaluació de resultats

Els sistemes d'avaluació de resultats són essencials per mesurar l'eficàcia dels models de reconeixement d'entitats nomenades. Aquests sistemes utilitzen diversos paràmetres per determinar la precisió, el reclam i la mesura F dels models (Nadeau i Sekine, 2007). Cadascuna d'aquestes mètriques funciona de la forma següent:

- Precisió. La precisió mesura la proporció d'identificacions correctes. Compta totes les instàncies correctes extretes pel model i les divideix entre el nombre total d'instàncies, tant les correctes com les incorrectes. El percentatge resultant indica quina quantitat d'elements recuperats són correctes.
- Reclam. El reclam avalua quin percentatge d'entitats reals ha estat correctament identificat pel model. Compta totes les instàncies correctes extretes pel model i les divideix entre el nombre total d'elements marcats manualment com a correctes. El percentatge resultant indica la quantitat d'instàncies correctes recuperades respecte del total.
- Mesura F. La mesura F és la mitjana harmònica entre precisió i reclam. Aquesta mètrica és important perquè proporciona una única mesura de rendiment.

Per a l'avaluació, s'utilitzen conjunts de dades anotats manualment com a referència. Aquests conjunts de dades els preparen lingüistes experts que revisen i anoten textos, assignant les etiquetes correctes a cada entitat. Els resultats són llavors comparats amb aquestes anotacions per determinar l'exactitud del model.

Existeixen molts models que utilitzen aquestes mètriques de diverses formes. Les més senzilles, anomenades avaluacions per coincidència exacta, es basen completament en les mètriques ja presentades. El sistema d'avaluació CONLL, que funciona d'aquesta manera, serà l'utilitzat per avaluar el model REN adaptat per a aquest treball.

Altres models més complicats alteren la mesura F final tenint en compte més paràmetres. Un exemple d'aquest model és el MUC, presentat a la conferència MUC-5 el 1993. Altres models com ACE segueixen un patró complex per poder tenir en compte moltes més variables. Tot i així, a causa de la seva complexitat se n'ha descartat l'ús per aquest treball.

Sigui com sigui, fer recerca amb un enfocament rigorós en l'avaluació és crucial per al desenvolupament de models de REN robustos i fiables que puguin ser utilitzats eficaçment en diverses llengües i contexts. D'aquesta manera, garanteixen una adaptació òptima en les aplicacions de processament de llenguatge natural.

Capítol 3

Metodologia

Per assolir l'objectiu d'aquest treball acadèmic, caldrà dividir el procés en diverses parts més petites. En tractar-se d'un estudi amb una alta càrrega tècnica, aquestes parts permetran analitzar més detalladament tots els aspectes que intervenen per obtenir el producte final. Si considerem un text d'entrada com a estadi inicial i un glossari d'entitats nomenades del mateix text en un arxiu importat a una eina de traducció assistida per ordinador com a estadi final, el procés es pot dividir en quatre parts.

3.1 Preparació del text amb la llibreria spaCy

La primera part és la preparació del text a partir del qual s'extrauran les entitats nomenades. Aquesta preparació inclou la tokenització, la desambiguació lèxica i el reconeixement d'entitats nomenades que és capaç de dur a terme la llibreria de Python spaCy.

Es tracta d'un procés totalment automàtic en dues parts. La primera, que inclou la tokenització i la desambiguació lèxica, s'efectua amb una precisió i reclam de pràcticament el 100%. La segona inclou el reconeixement d'ENAMEX i s'efectua amb una precisió i reclam d'aproximadament el 80%. El sistema utilitza un model d'aprenentatge automàtic per al català entrenat amb el corpus AnCora per identificar i classificar les entitats dins dels textos.

Per a aquesta primera part, i per a totes les següents, s'utilitzaran textos periodístics en català. D'una banda, això es deu a l'alt contingut d'entitats nomenades que contenen aquesta mena de textos, tant de noms propis com d'expressions temporals i numèriques. De l'altra, la llibreria spaCy i el corpus AnCora, es nodreixen principalment d'aquesta tipologia de textos.

3.2 Reconeixement d'entitats nomenades addicionals mitjançant regles

La segona part se centra en el reconeixement de TIMEX i NUMEX mitjançant regles. Això es podrà assolir mitjançant expressions regulars aplicades dins del codi mateix, posteriorment a l'aplicació del model d'aprenentatge automàtic de spaCy. Aquest pas és crucial perquè acaba de definir la qualitat de les dades que seran utilitzades en les etapes subsegüents del projecte.

Pel que fa a les ENAMEX, la dificultat de detecció d'aquest tipus d'entitats impedeix retocs al codi utilitzat. En basar-se totalment en la part d'aprenentatge automàtic, el marge per millorar-ne la detecció és molt menor.

3.3 Avaluació de les llistes obtingudes

Tot seguit, la tercera part del projecte es dedicarà a l'avaluació de les llistes d'entitats generades. Bàsicament, s'avaluaran les llistes mitjançant el sistema per coincidències exactes CONLL, com ja s'-ha comentat anteriorment. Les mètriques de precisió, reclam i mesura F, conjuntament amb un corpus anotat manualment amb aquesta finalitat, permetran analitzar detalladament quines entitats el model de REN detecta millor i per a quines té més dificultats.

3.4 Creació de les llistes importables i ús a l'eina TAO

Una vegada identificades les entitats, el següent pas és la creació d'una llista d'entitats que pugui ser importada a una eina de traducció assistida per ordinador (TAO) com MemoQ. El format de la llista serà compatible amb els estàndards de MemoQ per a la seva fàcil importació i ús posterior.

Aquesta etapa provarà la funcionalitat de les llistes en un entorn real de traducció, en què es podran utilitzar les entitats reconegudes per millorar la coherència i la qualitat d'una traducció.

Capítol 4

Implementació d'un model de REN híbrid

L'objectiu d'aquest capítol és la implementació d'un model de reconeixement d'entitats nomenades capaç de detectar els tres tipus d'entitats presentats: ENAMEX, TIMEX i NUMEX. En fer una comparació entre els models de REN per a l'anglès i per al català de la llibreria de Python spaCy, destaca la diferència entre els tipus d'entitats que pot trobar cadascun. El model per a l'anglès va ser entrenat per trobar tots tres tipus, mentre que el català es va limitar a la detecció d'ENAMEX.

Tots dos models utilitzen un conjunt d'etiquetes per obtenir uns resultats més precisos, tot i que el de català és molt més reduït. La Taula 4.1 detalla les etiquetes que utilitza cada model, juntament amb el tipus d'entitats a les quals s'assignen.

En vista d'aquestes etiquetes, el primer pas és l'obtenció de les entitats que el model en català ja és capaç de trobar. Tot seguit, es crearan unes regles mitjançant expressions regulars per obtenir totes aquelles etiquetes de TIMEX i NUMEX les quals el model no va ser entrenat per detectar. Això inclou les etiquetes 'CARDINAL', 'DATE', 'MONEY', 'ORDINAL', 'PERCENT', 'QUANTITY' i 'TIME'. Finalment, s'obtindrà la llista final amb totes les entitats nomenades.

Pel que fa a les etiquetes d'ENAMEX que no és capaç de detectar, la complexitat d'aquesta mena d'entitats limita la possibilitat de crear regles per a cadascuna. L'enfocament necessari, basat en la modificació del model d'aprenentatge automàtic utilitzat pel model en català de spaCy, s'escapa de l'abast d'aquest treball acadèmic.

Per tal d'identificar més fàcilment les entitats que es van reconeixent a mesura que avança la recerca, s'utilitzarà un fragment de text de contingut fictici creat amb aquesta finalitat. Aquest fragment contindrà

Etiqueta	Tipus d'entitat	Entitats que reconeix	Anglès	Català
'CARDINAL'	NUMEX	Qualsevol nombre que no sigui d'una altra categoria	Sí	No
'DATE'	TIMEX	Dates absolutes	Sí	No
'EVENT'	ENAMEX	Esdeveniments	Sí	No
'FAC'	ENAMEX	Edificis i equipaments	Sí	No
'GPE'	ENAMEX	Països i ciutats	Sí	Sí (a 'LOC')
'LANGUAGE'	ENAMEX	Glotònims	Sí	No
'LAW'	ENAMEX	Noms de lleis	Sí	No
'LOC'	ENAMEX	Qualsevol localització que no sigui d'una altra categoria	Sí	No
'MISC'	ENAMEX	Miscel·lània	No	Sí
'MONEY'	NUMEX	Monedes i divises	Sí	No
'NORP'	ENAMEX	Grups religiosos, nacionals o partits polítics	Sí	Sí (a 'ORG')
'ORDINAL'	NUMEX	Numerals ordinals	Sí	No
'ORG'	ENAMEX	Qualsevol organització que no sigui d'una altra categoria	Sí	Sí
'PERCENT'	NUMEX	Percentatges	Sí	No
'PER'	ENAMEX	Noms de persona, reals o ficticis	Sí	Sí
'PRODUCT'	ENAMEX	Productes per al comerç	Sí	No
'QUANTITY'	NUMEX	Unitats de mesura	Sí	No
'TIME'	TIMEX	Expressions de temps i dates relatives	Sí	No
'WORK_OF_ART'	ENAMEX	Obres d'art	Sí	No

Taula 4.1: Etiquetes assignades per cada model.

diversos exemples de cada etiqueta que s'estigui buscant. El primer fragment de text, separat per tokens i encara sense etiquetes, es mostra a la Figura 4.1.

Christine Lagarde , presidenta d el Banc Central Europeu , va anunciar l' 11 d' abril de 2024 a Frankfurt que no hi hauria una setena pujada d els tipus d' interès , que actualment es troben a l 4,5% . Va afirmar que les pujades de l' any passat eren suficients per frenar una inflació que , des d el 2020 , ha augmentat la despesa mitjana per habitatge en més de 100 € mensuals . Mentrestant , a uns 1.200 kilòmetres , unes 5.000 persones es manifestaven a Barcelona contra ...

Figura 4.1: Fragment de text sense etiquetar.

4.1 Identificació d'ENAMEX

Per obtenir el primer conjunt d'etiquetes ENAMEX, generades automàticament pel model de spaCy per al català, cal instal·lar el paquet corresponent. El model ofereix paquets de diverses mides: petit, mitjà i gran. Els paquets més grans obtenen millors resultats en precisió i reclam perquè van ser entrenats amb un corpus més extens, però per simplicitat i eficàcia s'ha escollit el més petit.

Un cop instal·lat el paquet, amb una sola línia de codi es pot demanar a aquesta llibreria que tokenitzi, executi el desambiguador lèxic i trobi totes les entitats nomenades que és capaç de detectar en un text. El resultat per al fragment de text d'exemple es mostra a la Figura 4.2. S'ha generat mitjançant una funció de la llibreria que permet visualitzar totes les entitats nomenades detectades.

Christine Lagarde PER , presidenta del Banc Central Europeu ORG , va anunciar l'11 d'abril de 2024 a Frankfurt MISC que no hi hauria una setena pujada dels tipus d'interès, que actualment es troben al 4,5%. Va afirmar que les pujades de l'any passat eren suficients per frenar una inflació que, des del 2020, ha augmentat la despesa mitjana per habitatge en més de 100 € mensuals. Mentrestant, a uns 1.200 kilòmetres, unes 5.000 persones es manifestaven a Barcelona LOC contra...

Figura 4.2: Fragment de text etiquetat amb ENAMEX.

Aquest primer resultat mostra un bon funcionament del model, tot i que amb algun error. Ha sigut capaç

d'assignar correctament les etiquetes 'PER' a *Christine Lagarde*, 'ORG' a *Banc Central Europeu* i 'LOC' a *Barcelona*. En canvi, no ha sigut capaç d'assignar correctament l'etiqueta 'LOC' a *Frankfurt*, al qual li ha assignat l'etiqueta 'MISC'.

Amb aquesta base, és el moment de començar a crear regles per trobar noves entitats nomenades.

4.2 Identificació de TIMEX

Per identificar el conjunt d'entitats TIMEX, que comprèn les etiquetes 'DATE' i 'TIME', s'han dissenyat tres expressions regulars. La primera se centra en l'etiqueta 'DATE' i està destinada a detectar totes les dates absolutes, independentment del format en què es presentin. La segona s'enfoca en l'etiqueta 'TIME', amb l'objectiu de trobar expressions de temps relatives, així com altres formes temporals. La tercera, també centrada en l'etiqueta 'TIME', està pensada per capturar expressions de freqüència.

Cada expressió regular ha estat dissenyada tenint en compte la variabilitat lingüística, assegurant-ne a més l'eficàcia per identificar aquestes entitats tant si apareixen al començament d'una frase —és a dir, si comencen per majúscula— com en qualsevol altra posició dins del text —és a dir, si comencen per minúscula. A continuació es presentaran detalladament cadascuna d'aquestes expressions regulars, destacant les seves característiques distintives i la forma en què s'han abordat els reptes particulars de cada tipus d'entitat TIMEX.

4.2.1 Etiqueta 'DATE'

El principal repte d'aquesta etiqueta és la quantitat i variabilitat a partir de les quals la llengua permet expressar dates absolutes i hores. Per tal d'incloure tota aquesta variabilitat, s'ha separat l'expressió regular en diversos patrons de cerca, cadascun centrat en trobar un tipus de data. L'expressió regular dissenyada que assigna l'etiqueta 'DATE' és la següent[1]:

```
regex_dates_hores = (r''
    r'\b(?:0?[1-9]|[12][0-9]|3[01])[-/](?:0?[1-9]|1[0-2])'
        r'([-/](\d{2}|\d{4}))?\b|'
    r'\b(?:0?[1-9]|[12][0-9]|3[01])\s(de\s|d\')(?:gener|febrer|març|abril|maig|juny|'
        r'juliol|agost|setembre|octubre|novembre|desembre)(\s(del?\s)(\d{2,4}))?\b|'
    r'\b(([Ee]l\s)?gener|([Ee]l\s)?febrer|([Ee]l\s)?març|([Ll]\')?abril|([Ee]l\s)?maig'
        r'|([Ee]l\s)?juny|([Ee]l\s)?juliol|([Ll]\')?agost|([Ee]l\s)?setembre|([Ll]\')?'
```

[1]Per a aquesta expressió regular i per a totes les següents, es distingirà cada patró de cerca amb un sagnat al principi de línia. Al seu torn, un doble sagnat significa que la línia segueix mostrant el patró de l'anterior.

```
    r'octubre|([Ee]l\s)?novembre|([Ee]l\s)?desembre)(\s(del?\s)(\d{2,4}))\b|'
r'\b([01]?\d|2[0-3])([:\.][0-5]\d\s?)(hores|h)?\b|'
r'\b(([Ee]l\s)|[Ll]\'any\s)?\d{4}\b|'
r'\b[Ss]egle\sM{0,3}(CM|CD|D?C{0,3})(XC|XL|L?X{0,3})(IX|IV|V?I{0,3})\b|'
r'\d{1,2}\-[GFMAJSOND]')
```

El primer patró se centra en trobar dates en xifres separades per barra (/) o guionet (-). Per als dies i els mesos amb un sol dígit, és capaç de detectar la xifra amb un zero o sense. Pel que fa als anys, d'expressió opcional, es poden expressar tant amb dues com amb quatre xifres. Per tant, pot detectar dates com *05/11/99*, *15-9-2006* o *17/04*.

El segon patró és capaç de reconèixer dates en què el dia i l'any s'expressen amb xifres i el mes s'expressa amb el nom complet. Si el mes comença per vocal, troba la preposició *de* apostrofada. En canvi, si comença per consonant, la troba completa. A més a més, l'any és d'expressió optativa. Com a resultat, és capaç de detectar dates com *19 de setembre de 1995*, *31 d'agost de 2022* o *15 de maig*.

Al seu torn, el tercer patró cerca dates en què només s'expressa el mes amb el nom complet i l'any en dues o quatre xifres. De forma opcional, capta l'article que precedeix el mes. El patró detecta dates com *gener de 2023* o *l'agost del 69*.

Seguidament, el quart patró reconeix hores exactes expressades amb xifres. Les hores i els minuts poden anar separats per un punt o per dos punts. La xifra sencera pot anar seguida d'espai i d'una hac o del mot *hores*. Per tant, detecta expressions com *14.50 h*, *22.00 hores* o *09:30*.

El cinquè patró cerca anys que s'expressen amb quatre xifres, amb la possibilitat d'estar precedits per article, pel substantiu *any* o per tots dos alhora. Aquest patró permet identificar expressions d'anys com ara *El 1895*, *l'any 2000* o *1789*.

El sisè patró és capaç de detectar segles representats en nombres romans en majúscules, precedits per la paraula *segle* que pot començar amb majúscula o minúscula. El patró permet trobar qualsevol combinació vàlida de nombres romans. D'aquesta manera, és capaç de detectar expressions de segles com *Segle XXI* o *segle LXI*.

Finalment, el setè patró s'encarrega de detectar dates destacades. Detecta un dia representat amb un o dos dígits, seguit d'un guionet i de la lletra inicial d'un mes en majúscula. Troba dates com *15-M* o *23-F*.

L'expressió regular aplicada a un fragment de text d'exemple, com mostra la Figura 4.3, és capaç de reconèixer les dates que hi apareixen sense dificultats.

Figura 4.3: Fragment de text amb l'etiqueta 'DATE'.

4.2.2 Etiqueta 'TIME'

De forma similar a l'anterior, el repte d'aquesta etiqueta es basa en la variabilitat d'expressions que expressen dates relatives i freqüències. D'una banda, les dates relatives representen totes aquelles expressions temporals que requereixen conèixer el dia actual per saber exactament a quin punt del temps es refereixen.

De l'altra, les freqüències representen punts temporals amb un patró repetitiu, no fixats a una data concreta. Per qüestions de claredat i simplicitat, s'han creat dues expressions regulars, una per a cada tipus d'expressió temporal. Totes dues assignen l'etiqueta 'TIME' a les entitats que troben dins del text.

Per trobar aquestes entitats nomenades, s'han de tenir en compte una gran diversitat de formats, els quals poden fer referència a dies de la setmana, mesos o anys. Això exigeix que l'expressió regular reconegui una àmplia gamma de patrons de cerca, cadascun amb la seva pròpia sintaxi. A més, cada patró s'ha de construir de manera que funcioni per separat però també en conjunt. Això és essencial per assegurar que expressions diferents com *l'any passat* o *el mes passat* no interfereixin entre si.

La primera expressió regular, dedicada a la cerca de dates relatives, es presenta a continuació:

```
regex_dates_relatives = (r''
    r'\b((?:[Aa]quest\s)?([Dd]illuns|[Dd]imarts|[Dd]imecres|[Dd]ijous|'
        r'[Dd]ivendres|[Dd]issabte|[Dd]iumenge)(?:\sque\sve|\spassat)?)\b|'
    r'\b([Aa]questa\ssetmana|[Aa]quest\smes|[Aa]quest\sany)\b|'
    r'\b(([Ll]a\ssetmana\s|[Ee]l\smes\s|[Ll]\'any\s)(que\sve|passa(t|da)|'
        r'precedent|següent|anterior|posterior))\b|'
    r'\b([Ff]a\s|[Dd]\'aquí\sa?\s?|[Cc]ada\s?)?([Gg]airebé\s|[Tt]ot\sjust\s|'
        r'([Pp]oc\s)?més\s(d\'|de))?([Uu]na?e?s?\s|[Dd](o|ue)s\s|[Tt]res\s|'
        r'[Qq]uatre\s|[Cc]inc\s|[Ss]is\s)(di(a|es)|setman(a|es)|meso?s?|anys?)\b|'
```

```
r'\b([Aa]vui|[Dd]emà|([Aa]bans-d\')?[Aa]hir|[Dd]emà\spassat)\b|'
r'\b((?:[Jj]ust\s)?[Aa]ra|(?:[Mm]olt\s|[Pp]oc\s|[Jj]ust\s)?[Aa]bans|'
    r'(?:[Pp]oc\s|[Jj]ust\s)?[Dd]esprés)\b|'
r'\b[AaEe]\s?l\'?\s(mes\s(de\s|d\'))?(gener|febrer|març|abril|maig|juny|juliol|'
    r'agost|setembre|octubre|novembre|desembre)\b|'
r'\b(((?:[Pp]rimera\s|[Úú]ltima\s)hora\s(?:del\s|de\sla\s))?|([Aa]quest\s|'
    r'[AaEe]l\s|[Ll]a\s|[Aa]\sla\s)?)(matí|migdia|tarda|vespre|nit|matinada)|'
r'\b([Ll]a|[Ll]es)\s(una|dues|tres|quatre|cinc|sis|set|vuit|nou|deu|onze|dotze|'
    r'\d{1,2}))(?:\s(del\smatí|del\smigdia|de\sla\starda|del\svespre|de\sla\snit|'
    r'de\sla\smatinada))\b')
```

El primer patró detecta dies de la setmana, tant en expressions absolutes com relatives. És capaç de trobar noms complets de dies de la setmana precedits opcionalment per demostratiu o seguits, també opcionalment, per expressions temporals de proximitat, tant de passat com de futur. Per exemple, és capaç d'identificar entitats com *dilluns*, *Aquest dimarts* o *diumenge que ve*.

Els dos patrons de cerca següents compleixen la mateixa funció, però especialitzades en trobar entitats que fan referència a períodes de temps més extensos. A més, inclou una gamma d'expressions temporals de proximitat més àmplia, pròpia d'aquests tipus d'expressions relatives. Així, detecta expressions com *aquesta setmana*, *el mes que ve* o *l'any precedent*.

El quart patró amplia la capacitat de detectar expressions dels tres anteriors. Es dedica a trobar expressions temporals de proximitat típiques dels textos periodístics com *fa gairebé dos anys*, *Fa tot just cinc dies* o *d'aquí poc més d'un mes*.

Seguidament, el cinquè patró identifica adverbis de temps relacionats amb la data actual. Això inclou adverbis com *avui*, *abans-d'ahir* o *Demà passat*. De forma similar, el sisè patró identifica adverbis de temps relacionats amb el moment actual. És capaç de trobar expressions com *Ara*, *molt abans* o *poc després*.

El setè patró reconeix referències al mes de l'any actual precedits d'article definit, amb contracció o sense. Opcionalment, també l'antecedeix el substantiu *mes* i la preposició *de*. D'aquesta manera, captura expressions com *al juny* o *El mes d'agost*.

Tot seguit, el vuitè patró és capaç de detectar referències a parts del dia, siguin precedides de determinant o de demostratiu o siguin precedides d'una locució que concreta encara més el moment temporal. Per tant, troba expressions com *aquesta matinada* o *A primera hora de la tarda*.

Finalment, el novè patró és capaç de reconèixer hores concretes indicades amb paraules seguides per

referències a parts del dia. Així, és capaç d'identificar expressions com *La una de la matinada* o *les vuit del vespre*.

La segona expressió regular, centrada en cercar expressions de freqüència, és la següent:

```
regex_frequencies = (r''
    r'\b(([Cc]ada|([Uu]na?|[Dd]os|[Dd]ues|[Tt]res|[Qq]uatre|[Cc]inc|[Ss]is|'
        r'[Ss]et|[Vv]uit|[Nn]ou|[Dd]eu))\s(cops?|vegad(a|es)\s(al\s|a\sla\s|'
        r'(a\s)?l\'|per\s))(dia|setmana|mes|any|estiu|tardor|hivern|'
        r'primavera|dilluns|dimarts|dimecres|dijous|divendres|dissabte|diumenge)\b|'
    r'\b[Ee]ls\s(dilluns|dimarts|dimecres|dijous|divendres|dissabte|diumenge)\b|'
    r'\b([Dd]i[aà]ria?|[Ss]etmanal|[Qq]uinzenal|[Pp]eriòdica?|'
        r'[Ff]reqüent|[Oo]casional|([Bb]i|[Tt]ri|[Qq]uadri|[Ss]e)?([Mm]ensuals?|'
        r'mestral|[Aa]nuals?|ennal))(ment)?)\b|'
    r'\b[Ss]empre|[Mm]ai|[Ss]ovint|(([Aa]|[Dd]e)\svegades)|[Dd]e\stant\sen\stant\b')
```

El primer patró de cerca detecta entitats que indiquen una freqüència basada en repeticions. Han de començar pel determinant *cada* o per un numeral de l'u al deu. A més, els numerals han d'anar acompanyats pel substantiu *cop* o *vegada*. Després, ha d'aparèixer una unitat de temps específica. Pot capturar expressions com *Cada estiu*, *dues vegades a la setmana* o *tres cops l'any*.

El segon patró reconeix expressions simples de freqüència associades als dies de la setmana que apareixen en plural com, per exemple, *Els dilluns*.

El tercer patró detecta adjectius de freqüència relacionats amb la periodicitat. A més, els adjectius de freqüència específics per als mesos i els anys poden portar annexats un prefix quantificador. Tot el conjunt d'adjectius pot anar seguit del sufix adverbialitzador *-ment*. Així, és capaç de detectar expressions com *Ocasionalment*, *trimestral* o *biennal*.

L'últim patró s'encarrega de trobar altres expressions de freqüència simples com *sempre*, *Mai* o *de tant en tant*.

El resultat d'aplicar aquestes expressions regulars al fragment de text d'exemple es pot observar a la Figura 4.4. Cada expressió regular ha detectat diverses entitats nomenades, les quals apareixen etiquetades correctament a l'exemple.

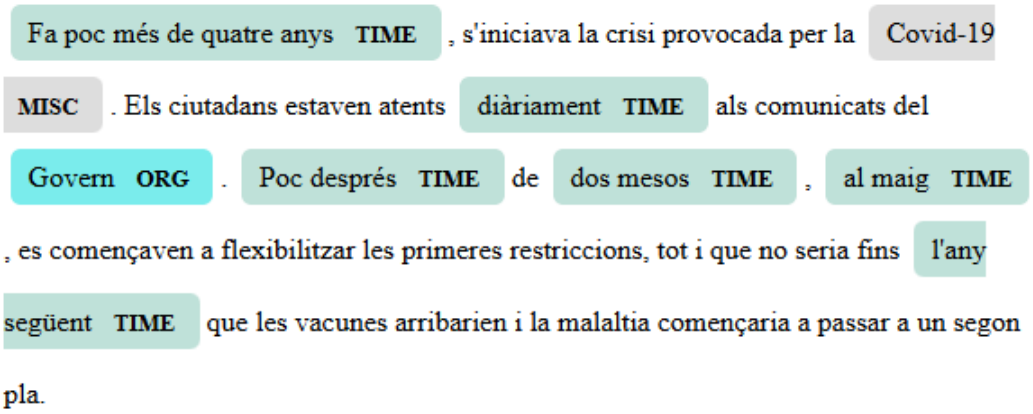

Figura 4.4: Fragment de text amb l'etiqueta 'TIME'.

4.3 Identificació de NUMEX

Per a la detecció del conjunt d'entitats nomenades de tipus NUMEX, format per la resta d'etiquetes, s'ha formulat una expressió regular per a cadascuna. Cal tenir en compte que el llenguatge natural és molt ampli i presenta una gran variabilitat en l'ús de xifres i paraules per expressar conceptes numèrics. Aquesta variabilitat es reflecteix en diferents formats, unitats i combinacions que poden ser presents en els textos.

Així mateix, és important considerar les ambigüitats i els diferents significats que poden tenir les expressions numèriques en diferents contextos. Per exemple, un nombre pot fer referència a una unitat monetària, a una unitat de mesura o a qualsevol altre tipus d'element. Aquesta complexitat fa que el disseny de les expressions regulars sigui un repte i requereixi una atenció especial per a cada etiqueta. En formular aquestes expressions regulars, s'han tingut en compte aquests factors per garantir-ne robustesa i capacitat i poder identificar correctament els diferents tipus de NUMEX.

4.3.1 Etiqueta 'ORDINAL'

L'expressió regular que assigna l'etiqueta 'ORDINAL' s'ha dissenyat per identificar les diferents formes d'ordinals en català. Detecta tant les variants en lletres com en nombres romans i xifres. Aquesta expressió s'ha desglossat en tres patrons de cerca per cobrir tota la variabilitat. L'expressió regular es presenta a continuació:

```
regex_ordinals = (r''
    r'\b([Pp]rimer(a|e?s)?|[Ss]egon(a|es)?|[Tt]ercer(a|e?s)?|[Qq]uart(a|e?s)?|'
        r'[Cc]inqu[èe](ns|na|nes)?|[Ss]is[èe](ns|na|nes)?|([Dd]is)?[Ss]et[èe]'
        r'(ns|na|nes)?|([Dd]i)?[Vv]uit[èe](ns|na|nes)?|([Dd]i)?[Nn]ov[èe]'
```

```
    r'(ns|na|nes)?|[Dd]es[èe](ns|na|nes)?|[Oo]nz[èe](ns|na|nes)?|[Dd]otz[èe]'
    r'(ns|na|nes)?|[Tt]retz[èe](ns|na|nes)?|[Cc]atorz[èe](ns|na|nes)?|'
    r'[Qq]uinz[èe](ns|na|nes)?|[Ss]etz[èe](ns|na|nes)|[Vv]int[èe](ns|na|nes)??)\b|'
r'\b\d+(rs?|ts?|è|a|ns?|es)\.?\b|'
r'\b(?=[MDCLXVI])M{0,3}(CM|CD|D?C{0,3})(XC|XL|L?X{0,3})(IX|IV|V?I{0,3})'
    r'\b(?=\s[A-Z])')
```

El primer patró detecta les formes en lletres dels ordinals de l'u al vint. Tots els ordinals inclouen flexions de gènere i nombre, excepte en el cas del plural masculí de *segon*. Això es deu a la freqüència amb la qual s'utilitza la conjunció o preposició *segons* en textos periodístics. Si s'inclogués en el patró de cerca, hi hauria molts falsos positius. Com a resultat, el patró troba ordinals com *Primer*, o *vuitenes*.

El segon patró cerca ordinals expressats en nombres i inclou flexions de gènere i nombre. A més, els ordinals poden portar punt després de la lletra. Per tant, és capaç de detectar ordinals com *5a*, *9è* o *35ns.*.

L'últim patró de l'expressió regular està dissenyat per reconèixer ordinals en nombres romans, els quals sovint precedeixen esdeveniments o títols. El patró exigeix que després del nombre romà hi hagi un espai seguit d'una paraula que comenci amb majúscula. Això evita que es detectin xifres romanes que s'utilitzen en altres contextos, com els segles. Així doncs, el patró detecta ordinals en xifres romanes com *XXIV*.

La Figura 4.5 mostra com s'han aplicat correctament dues etiquetes 'ORDINAL'. Tot i això, l'expressió regular no ha aplicat l'etiqueta a l'ordinal en nombres romans. Això es deu al fet que la llibreria de spaCy reconeix aquesta entitat amb l'etiqueta 'MISC' i, per tant, no se li pot assignar una segona etiqueta. Aquesta situació evidencia una de les limitacions intrínseques associades a l'ús de models híbrids en el processament de llenguatge natural.

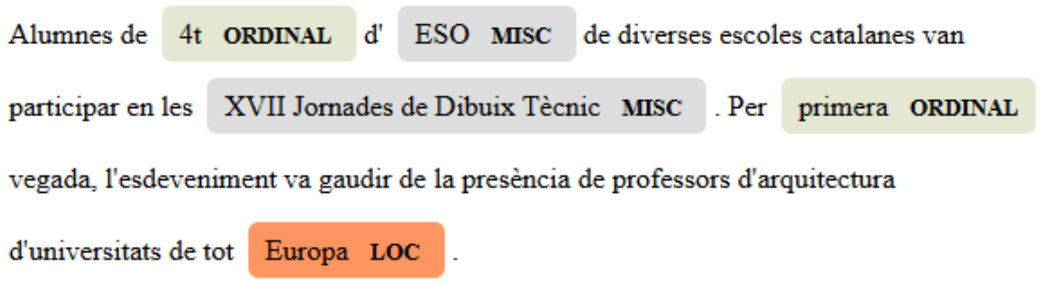

Figura 4.5: Fragment de text amb l'etiqueta 'ORDINAL'.

4.3.2 Etiqueta 'MONEY'

L'expressió regular que assigna l'etiqueta 'MONEY' s'utilitza per identificar quantitats de diners expressades en diverses monedes i en diferents formats. A diferència de les expressions regulars anteriors, consta d'un únic patró de cerca que, tot i ser molt extens, simplifica considerablement la cerca d'entitats nomenades. L'expressió regular és la següent:

```
regex_monedes = (r''
    r'(\d{1,3}(?:,\d{1,3})?\s?(\.\d{3})*(?:,\d{1,3})?\s?)(mil(ers)?\s(de\s)?)'
        r'?([bm]ilions\s(de\s)?)?((cèntims?|centaus?|¢|penics?)|(afganis?|AFN)'
        r'|(ariarys?|MGA)|(฿|bahts?|THB)|(balboas?|PAB)|(birrs?|ETB)|'
        r'(bolívars?|VEF)|(bolivianos?|BOB)|(cedis?|GHS)|(₵|colons?|CRC)|'
        r'(córdob(a|es)|NIO)|(coron(a|es)|DKK|SKK|EEK|ISK|NOK|SEK|CZK)|'
        r'(dalasis?|GMD)|(denars?|MKD)|(dinars?|DZD|BHD|IQD|JOD|KWD|LYD|TND|'
        r'RSD|IRR)|(dírhams?|AED|MAD)|(dobr(a|es)|STD)|(\$|\$CAN|dòlars?'
        r'|AUD|BSD|BBD|BZD|BMD|BND|KYD|CAD|XCD|USD|FJD|GYD|HKD|JMD|LRD|NAD|NZD'
        r'|SBD|SGD|SRD|TWD|TTD|ZWL)|(₫|dongs?|VND)|(DR|dracm(a|es))|(drams?|AMD)|'
        r'(escuts?|CVE)|(M?€|d?\'?euros?|M?EUR)|(f|FL|flor(í|ins)|ANF|AWG)|'
        r'(fòrints?|HUF)|(FB|FLUX|FS|francs?|BIF|XAF|XOF|XPF|KMF|CDF|DJF|GNF|'
        r'RWF|CHF)|(gourdes?|HTG)|(guaranís?|PYG)|(hrívni(a|es)|UAH)|'
        r'(¥|iens?|iuans?|renminbis?|JPY|CNY)|(kin(a|es)|PGK)|(kips?|LAK)|'
        r'(kun(a|es)|HRK)|(kwach(a|es)|MWK|ZMK)|(kwanz(a|es)|AOA)|(kyats?|MMK)|'
        r'(laris?|GEL)|(lats|LVL)|(leks?|ALL)|(L|lempir(a|es)|HNL)|(leones?|SLL)|'
        r'(leus?|MDL|RON)|(levs?|BGN)|(lilangenis?|SZL)|([ff]|lir(a|es)|TRY)|'
        r'(litas|LTL)|([ff]|£EG|£IR|£LIB|£SYR|£TQ|(lliur(a|es)(\sesterlin(a|es))?'
        r'|EGP|GBP|GIP|GGP|JEP|LBP|FKP|IMP|SHP|SYP|SDG|SSP))|(lotis?|LSL)|'
        r'(manats?|AZN|TMT)|(DM|MF|marcs?|BAM)|(meticals?|MZN)|(₦|nair(a|es)|NGN)|'
        r'(nafk(a|es)|ERN)|(ngultrums?|BTN)|(ouguiy(a|es)|MRO)|(pa\'ang(a|ues)|TOP)|'
        r'(pata(ca|ques)|MOP)|(₱|$MEX|$CH|CU|pesos?|ARS|COP|CUC|CUP|DOP|PHP|MXN|'
        r'UYU|CLP)|(pesset(a|es)|pt(a|es)\.?|ESP|M?PTA)|(pul(a|es)|BWP)|'
        r'(quetzals?|GTQ)|(rands?|ZAR)|(R$|reals?|BRL)|(rials?|YER|IRR|OMR)|'
        r'(riels?|KHR)|(ringgits?|MYR)|(riyals?|QAR|SAR)|(rubles?|BYR|RUB)|'
        r'(rupi(a|es)|INR|IDR|MVR|MUR|NPR|PKR|SCR|LKR)|(sols?(?=\s)|PEN)|'
        r'(soms?|KGS|UZS)|(somonis?|TJS)|(sucres?)|(tak(a|es)|BDT)|(tal(a|es)|WST)|'
        r'(tengu?es?|KZT)|(tögrögs?|MNT)|(tolars?)|(vatus?|VUV)|(₩|wons?|KPW|KRW)|'
        r'(£IS|(nous?\s)?xéquels?|ILS)|(xílings?|KES|SOS|TZS|UGX)|(zaires?)|'
        r'(ZL|z[␣ll]otys?|PLN)|¤)')
```

Tot i la longitud d'aquesta expressió regular, el seu funcionament és força senzill. Primer, cerca una xifra d'entre un i tres dígits. A continuació, de forma opcional, cerca una coma per a nombres decimals o un punt com a separador tantes vegades com sigui necessari per detectar milers, milions, etcètera. També de manera opcional, detecta substantius quantificadors com *milers* o *milions*. Per últim, troba la moneda en qüestió, sigui un símbol, sigui el nom complet o sigui l'identificador estandarditzat segons l'ISO 4217. La longitud d'aquesta expressió regular es deu al fet que conté els noms i identificadors de totes les monedes del món. Així doncs, l'expressió regular és capaç de detectar expressions numèriques monetàries com *27.000.000 dòlars*, *60,5 milions de pessetes* o *15.000 EUR*.

La Figura 4.6 mostra com l'expressió regular detecta correctament dues de les xifres monetàries que apareixen a l'exemple. De nou, tot i reconèixer *GBP* com a l'identificador monetari de les lliures, la presència d'una etiqueta prèvia del model entrenat per aprenentatge automàtic impedeix l'assignació d'una etiqueta nova.

El Ministeri d'Economia **ORG** ha confirmat que les reserves de l'or del país estan valorades en 50.000.000 dòlars **MONEY** , mentre que les importacions d'electrònica des de Corea del Sud **LOC** han superat els 3.000 milions de wons **MONEY** . A nivell domèstic, els preus de la cistella bàsica s'han mantingut estables. Les principals matèries primeres s'han venut per uns 100 € **MONEY** per unitat i els productes d'importació han superat les 150 GBP **ORG** per lot.

Figura 4.6: Fragment de text amb l'etiqueta 'MONEY'.

4.3.3 Etiqueta 'PERCENT'

L'expressió regular que assigna l'etiqueta 'PERCENT' identifica percentatges tant en forma numèrica com escrita. Es tracta d'una expressió regular senzilla que consta de dos patrons de cerca breus. L'expressió regular és la següent:

```
regex_percentatges = (r''
    r'\-?(?:\d{1,3}(?:\.\d{3})*|\d+)(?:,\d+)?\s?(%|per\scent)|'
    r'\b(?:zero|u|dos|tres|quatre|cinc|sis|set|vuit|nou|deu|onze|dotze|tretze|'
        r'catorze|quinze|setze|disset|divuit|dinou|vint|trenta|quaranta|'
        r'cinquanta|seixanta|setanta|vuitanta|noranta|cent)\sper\scent\b')
```

El primer patró està dissenyat per detectar nombres seguits del símbol %. Pot reconèixer nombres amb o sense separadors de milers, amb decimals o sense, i nombres negatius. D'aquesta manera, pot detectar expressions com *23%*, *-12,5%* o *1.500%*.

El segon patró de cerca és capaç de detectar percentatges escrits en paraules, seguit dels mots *per cent*. Cerca els nombres del zero al dinou i, a partir del vint, cada desena fins arribar al cent. Per tant, pot trobar percentatges com *zero per cent*.

La Figura 4.7 mostra el correcte etiquetatge d'aquests dos patrons a un fragment d'exemple.

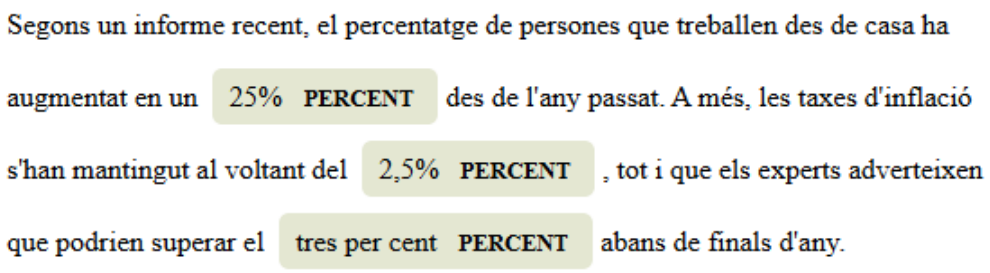

Figura 4.7: Fragment de text amb l'etiqueta 'PERCENT'.

4.3.4 Etiqueta 'QUANTITY'

L'etiqueta 'QUANTITY' s'assigna a unitats de mesura de qualsevol mena, sempre que vagin precedides per un nombre. És capaç de detectar les set unitats de mesura principals del Sistema Internacional d'Unitats, les unitats derivades, i altres unitats de mesura com les del sistema anglosaxó. Consta de dos patrons de cerca, cadascun centrat en trobar una forma concreta d'expressar aquestes unitats de mesura. L'expressió regular és la que segueix:

```
regex_mesures = (r''
    r'\b((-?\d{1,3}(?:[.,]\d{3})*(?:,\d+)?\s*))\s?
    r'((([QRYZEPTGMKkhdcmµnpfzyrq]|da)?'
        r'((m|g|s|A|K|mol|cd|N|Pa|J|W|C|V|S|F|T|Wb|H|rad|sr|Hz|lm|lx|Bq|Gy|Sv|'
        r'kat|B|b|L|l|cal|fg|mi|UA|ly|pc|t|HP|hp|ac|Ac|[ºª]C|[ºª]F|P|bar|atm)¯?'
        r'[¹²³123]?([p\·\/]?([QRYZEPTGMKkhdcmµnpfazyrq]?'
        r'(m|g|s|A|K|mol|cd|h)¯?[¹²³123]?))*))|
    r'\b([Qq]uett[aà]|[Rr]onn[aà]|[Yy]ott[aà]|[Zz]ett[aà]|[Ee]x[aà]|[Pp]et[aà]|'
        r'[Tt]er[aà]|[Gg]ig[aà]|[Mm]eg[aà]|[KkQq]u?il[oò]|[Hh]ect[oò]?|[Dd]ec[aà]|'
        r'[Dd]ec[ií]|[Cc]ent[ií]|[Mm]il·l[ií]|[Mm]icr[oò]|[Nn]an[oò]|[Pp]ic[oò]|'
        r'[Ff]emt[oò]|[Aa]tt[oò]|[Zz]ept[oò]|[Yy]oct[oò]|[Rr]ont[oò]|[Qq]ect[oò])?'
```

```
r'(metres?|grams?|segons?|minuts?|hor(a|es)|amperes?|(graus?\s)kelvin|'
r'candel(a|es)|mols?|newtons?|pascals?|joules?|watts?|coulombs?|volts?|'
r'ohms?|siemens?|farads?|tesl(a|es)|webers?|henrys?|radiant?s?|'
r'estereoradiant?s?|hertzs?|lumens?|luxs?|becquerels?|grays?|sieverts?|'
r'katals?|bels?|bytes?|bits?|litres?|gal[óo](ns)?|calori(a|es)|'
r'frigori(a|es)|mill(a|es)(\snàuti(ca|ques))?|llegu(a|es)|iard(a|es)|'
r'peus?|polzad(a|es)|bra(ça|ces)|unitats?\sastronòmi(ca|ques)|'
r'anys?\sllum|pàrsecs?|ton(a|es)|tonelad(a|es)|un(ça|ces)|cavalls?|'
r'àre(a|es)|acres?|graus?|graus?\s(Celsius|centígrads?'
r'|Fahrenheits?)|poises?|bars?|atmosfer(a|es)|psis?)(\sper)?\s?'
r'(metres?|grams?|segons?|amperes?|graus?\skelvin|candel(a|es)|mols?|'
r'hor(a|es))?((al)?\squadrats?|al\scub|cúbics?)?)\b')
```

La primera part d'aquesta expressió regular és comuna per als dos patrons de cerca. Troba xifres positives o negatives, amb un punt de separador si fa falta i una coma per als decimals.

Tot seguit, el primer patró de cerca és capaç de detectar unitats de mesura en format abreviat. Cerca un prefix quantificador si apareix, seguit de la unitat en qüestió. Pot detectar les mesures tant si estan expressades per combinacions de les set unitats bàsiques com si estan expressades en unitats derivades. A més, considera l'aparició de barra (/) per a unitats compostes com *km/h*.

El segon patró cerca les mateixes unitats, però expressades amb el nom sencer. Realitza la cerca de manera similar a l'anterior, tenint en compte els prefixos quantificadors que precedeixen les unitats. També és capaç de detectar unitats compostes com *metres per segon al quadrat*.

La Figura 4.8 mostra un fragment de text correctament etiquetat basant-se en l'expressió regular presentada.

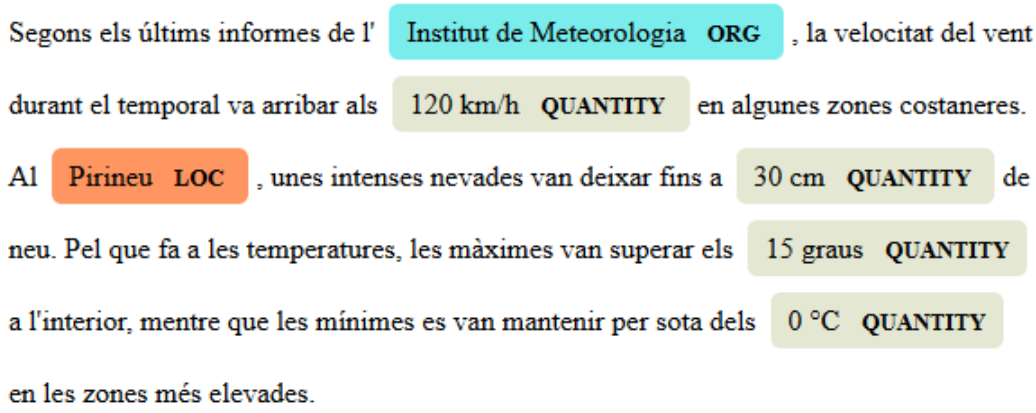

Figura 4.8: Fragment de text amb l'etiqueta 'QUANTITY'.

4.3.5 Etiqueta 'CARDINAL'

L'última etiqueta, 'CARDINAL', està centrada en trobar tota la resta de xifres que apareixen en un text. Cerca una xifra i el substantiu que la segueix. L'expressió regular dissenyada per trobar entitats amb aquesta etiqueta és la següent:

```
regex_quantitats = r''
    r'(\d+(?:[.,]\d{3})*(?:,\d+)?\s*)((?!(?:a|amb|en|entre|per|sense|'
        r'sota|sobre|del?|pel|al|i|o|que)\s)[^\s,\.\)\%\;]+)(?=\s|,|\.)'
```

Aquesta última expressió regular presenta una estructura notablement més senzilla en comparació amb les expressions regulars anteriors. Bàsicament, cerca una xifra seguida d'una paraula i aplica l'etiqueta a tot el conjunt. Si la xifra va seguida de preposició, contracció o conjunció, només captura els dígits. La Figura 4.9 mostra un fragment de text correctament etiquetat seguint el patró d'aquesta expressió regular.

Pel que fa a infraestructures, el nou pont que s'està construint sobre el riu oferirà 4 carrils CARDINAL addicionals per reduir la congestió del trànsit. A més, l'últim projecte d'energia renovable proporcionarà energia per a 200.000 llars CARDINAL , fet que augmentarà la capacitat de la xarxa nacional. També s'ha invertit en noves línies de metro, que esperen transportar uns 500.000 viatgers CARDINAL diaris.

Figura 4.9: Fragment de text amb l'etiqueta 'CARDINAL'.

4.4 Obtenció de les llistes

Fins ara, totes les expressions regulars presentades s'han mostrat en funcionament de forma aïllada a la resta. L'únic factor que podia afectar el resultat era la prèvia presència d'etiquetes assignades pel model d'aprenentatge automàtic. No obstant, el fet de processar un text amb diverses expressions regulars alhora pot comportar resultats inesperats. Per tal d'evitar errors potencials, s'ha establert un ordre de prioritats perquè algunes etiquetes s'assignin a una entitat nomenada abans que altres en cas de conflicte.

Abans de convertir les cadenes de text detectades per les expressions regulars en entitats nomenades, mitjançant codi s'ha creat una variable llista que emmagatzema tots els candidats. Tot seguit, es comparen tots els candidats entre si i, en cas de trobar dos candidats que comparteixen com a mínim un token,

s'elimina de la llista el que té menys prioritat.

Per a algunes etiquetes, la correcta atribució d'una prioritat és crítica. El cas més evident d'aquesta situació és el de l'etiqueta 'CARDINAL', a la qual se li ha assignat la prioritat més baixa. En cas de situar-la per davant d'altres, especialment d'etiquetes NUMEX, trobaria entitats nomenades que no li corresponen, com unitats de mesura o expressions monetàries amb el substantiu *milions*. A més, per la manera en què estan dissenyades les expressions regulars, el patró de cerca de l'etiqueta 'CARDINAL' s'apropia dels dígits dels anys expressats en xifres, tot i que no tinguin un punt com a separador de milers. Per tant, també és important que tingui una prioritat més baixa que les etiquetes TIMEX.

La resta de posicions en la llista de prioritat no son tan crucials, però ajuden a evitar alguns petits errors. Per exemple, entre les etiquetes 'TIME' i 'DATE', hi ha conflictes quan apareix el mot *maig*. L'etiqueta 'DATE' s'assigna correctament a tot el mot, mentre que l'etiqueta 'TIME' s'intenta assignar únicament als caràcters *mai* per la inclusió d'aquest mateix adverbi a l'expressió regular de freqüències. Per tant, per evitar aquesta situació, l'etiqueta 'TIME' ha de tenir una prioritat més baixa que l'etiqueta 'DATE'.

La llista de prioritats, declarada en una variable, està ordenada de major a menor prioritat. L'etiqueta 'PERCENT' té la màxima prioritat, seguida per 'MONEY', 'DATE', 'QUANTITY', 'ORDINAL', 'TIME' i, finalment, 'CARDINAL', tal com s'ha indicat anteriorment.

Com a últim exemple, la Figura 4.10 mostra el text presentat a l'inici d'aquest capítol després d'aplicar-hi totes les expressions regulars alhora. També s'han tingut en compte les prioritats de cada etiqueta.

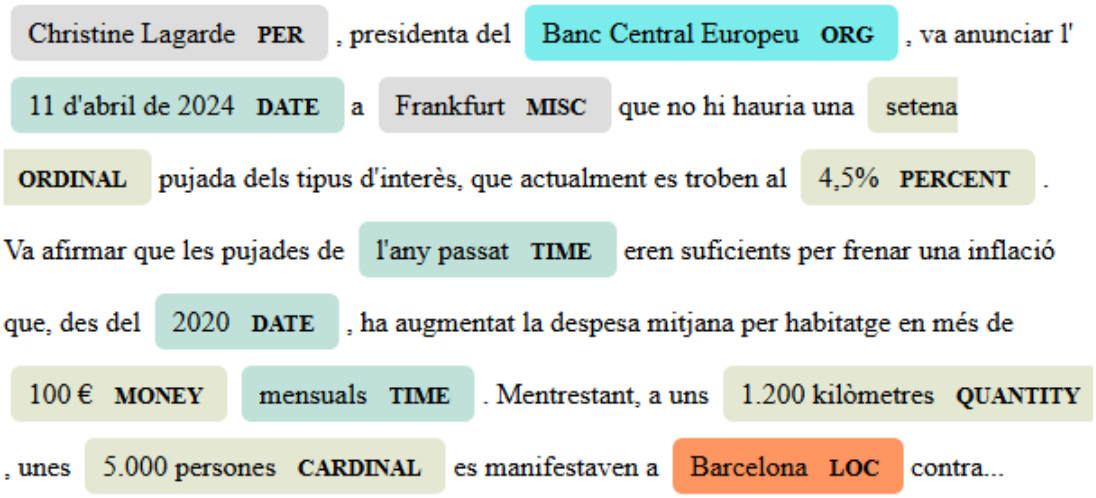

Figura 4.10: Fragment de text amb totes les etiquetes aplicades.

Un cop detectades i prioritzades correctament totes les entitats anomenades, ja es pot crear una llista que inclogui totes les deteccions. En funció del text d'entrada, aquesta llista servirà per a l'avaluació del model o per a l'eina de traducció assistida.

Capítol 5

Anàlisi dels resultats

L'anàlisi de resultats és un procés essencial durant el desenvolupament de models que operen dins del marc teòric del processament del llenguatge natural. Aquesta etapa no només valida la precisió i l'eficàcia del model en aplicar els principis pràctics i teòrics, sinó que també identifica aspectes susceptibles a una millora. Per al cas del model híbrid implementat en aquest treball, l'avaluació permet comprovar que es comporta de manera òptima en el context lingüístic de treball —els textos periodístics— i en diverses situacions d'ús real.

Per aprofundir encara més en l'optimització d'un model, els processos d'avaluació inclouen una fase d'afinament o *fine-tuning* durant la qual es revisen i ajusten repetidament els paràmetres que constitueixen el model. En el cas del model implementat, això implica fonamentalment modificacions menors a les expressions regulars. Les expressions regulars presentades en el capítol anterior ja han passat per aquest procés d'afinament.

Els resultats de l'avaluació es mostren en aquest capítol, juntament amb una anàlisi detallada del rendiment de cada etiqueta. L'avaluació s'ha dut a terme mitjançant l'ús del mateix corpus de proves que es va utilitzar per avaluar el model de spaCy en català, el qual està basat en el corpus d'AnCora (Taule et al., 2008). D'una banda, s'ha aplicat el model implementat al text del corpus, a partir del qual s'ha obtingut una llista d'entitats nomenades. De l'altra, s'ha completat el corpus manualment per incloure les noves etiquetes que s'intenta detectar. El corpus actualitzat està inclòs al repositori d'aquest treball de final de grau (Jacas, 2024).

Aquest corpus de prova, que consta d'unes 17.000 paraules, està digitalitzat mitjançant el format *.conllu*, utilitzat àmpliament per lingüistes computacionals per treballar amb textos etiquetats a diversos nivells lingüístics. El format *.conllu* facilita la manipulació i l'anàlisi detallada dels textos i proporciona una base sòlida per a l'avaluació rigorosa del model.

5.1 Avaluació del model de REN desenvolupat

Per tal de calcular les tres mètriques d'avaluació descrites al segon capítol —precisió, reclam i mesura F—, abans és necessari determinar tres tipus de resultats. Els primers són els autèntics positius (AP), els quals s'obtenen quan l'entitat nomenada detectada pel model i l'entitat nomenada anotada al corpus coincideixen totalment. En segon lloc, els falsos positius (FP) ocorren quan el model identifica una entitat nomenada que no està anotada al corpus. Per últim, els falsos negatius (FN) ocorren quan el model no ha detectat una entitat que apareix al corpus anotat manualment.

Després de comparar el corpus anotat manualment i els resultats del model implementat, s'han detectat 1.333 autèntics positius, 288 falsos positius i 296 falsos negatius. Si s'aplica la fórmula per a cada mètrica, els resultats són els següents:

$$Precisió = \frac{AP}{AP + FP} * 100 = \frac{1.333}{1.333 + 288} * 100 = 82,23\%$$

$$Reclam = \frac{AP}{AP + FN} * 100 = \frac{1.333}{1.333 + 296} * 100 = 81,82\%$$

$$Mesura\ F = 2 * \frac{Precisió * Reclam}{Precisió + Reclam} = 2 * \frac{82,23 * 81,82}{82,23 + 81,82} = 82,03\%$$

Per tant, el model implementat, després de cercar tant les entitats nomenades que ja era capaç de detectar com les trobades mitjançant expressions regulars, ha obtingut una mesura F del 82,03%. És una xifra relativament elevada que demostra el bon rendiment del model implementat. No obstant això, aquesta xifra no permet distingir entre les entitats detectades pel model inicial i les detectades per les expressions regulars.

En comparar les entitats trobades exclusivament pel model de spaCy i el corpus anotat amb totes les etiquetes, es van registrar 829 autèntics positius, 233 falsos positius i 800 falsos negatius. Això va resultar en una precisió del 78,06%, un reclam del 50,89% i una mesura F del 61,61%. La poca diferència en precisió es deu a la proporció similar de falsos positius i autèntics positius entre les dues avaluacions. En canvi, la diferència de més del 30% de reclam es deu als falsos negatius que en el model final esdevenen autèntics positius, fet que evidencia el bon rendiment de les expressions regulars.

5.2 Anàlisi detallada

Tot i els bons resultats obtinguts fins ara, la millor manera de demostrar l'eficàcia del model final és mitjançant una anàlisi desglossada per etiquetes. Juntament amb les tres mètriques principals, la detecció d'exemples concrets que s'han classificat com falsos positius o falsos negatius permetran discernir a quins problemes s'enfronta cada etiqueta i quines possibles solucions podrien millorar-ne el rendiment.

Respecte de la primera etiqueta, 'DATE', s'han detectat 109 autèntics positius, 1 fals positiu i 5 falsos negatius. Això implica un 99,09% de precisió, un 95,61% de reclam i una mesura F del 97,32%. El fals positiu es tracta de l'aparició d'una fracció, *2/4*, difícilment separable d'expressions en format de dia i mes separats per una barra. Entre els falsos positius s'hi troben expressions com *20h*, les quals resulten difícils de distingir d'expressions de mesura en cas de no tenir un espai separador; *del 4 al 7 d'octubre*, en què la primera xifra és difícil de distingir si no es busca l'expressió completa; o *a principis dels 90*, una expressió que no s'ha tingut en compte a l'expressió regular i que s'hi podria afegir fàcilment.

Pel que fa a la segona etiqueta, 'TIME', s'han detectat 162 autèntics positius, 36 falsos positius i 25 falsos negatius. Aquests resultats es tradueixen en un 81,81% de precisió, un 86,63% de reclam i una mesura F del 84,15%. Els falsos positius inclouen mots com *diari*, els quals sovint fan referència al substantiu i no a l'expressió de freqüència; expressions com *sis anys*, detectades per l'expressió regular que busca dates relatives i en la qual no es van tenir en compte aquesta mena d'expressions quan no van acompanyades d'expressions temporals com *més tard*, o *abans*; o mots com *dilluns* quan formen part d'una expressió temporal més concreta com *dilluns a la nit*. Els falsos negatius inclouen *començaments de juliol* o *aquesta nit*, expressions que no es van incloure en les expressions regulars.

Per a la tercera etiqueta, 'ORDINAL', s'han detectat 28 autèntics positius, 6 falsos positius i cap fals negatiu. Ha obtingut un 82,35% de precisió, un 100% de reclam i una mesura F del 90,32%. Els falsos positius es deuen a ordinals que apareixen en dates assenyalades com *Primer de Maig* o a numerals col·lectius com *una vintena*, difícilment distingibles mitjançant expressions regulars. D'altra banda, la detecció del numeral *onze* indica que l'expressió regular utilitzada per a aquesta etiqueta no distingeix correctament entre *onzè*, *onzena* i *onze* a causa de l'opcionalitat de l'accent per detectar aquest ordinal. Per millorar la precisió d'aquesta etiqueta, s'hauria de revisar la forma en què es detecten ordinals que inclouen accent per evitar que capturi numerals cardinals.

Quant a la quarta etiqueta, 'MONEY', s'han detectat 38 autèntics positius, cap fals positiu i cap fals negatiu. Això implica un 100% en precisió i reclam i una mesura F del 100%. Els bons resultats d'aquesta etiqueta indiquen que està ben acotada, tant pel que detecta com pel que evita detectar. Tot i això, en textos amb molta més càrrega d'expressions monetàries es podrien comprovar més acuradament les mancances de l'expressió regular que cerca aquest tipus d'entitat nomenada.

Pel que fa a la cinquena etiqueta, 'PERCENT', s'han detectat 58 autèntics positius, cap fals positiu i cap fals negatiu. De nou, això implica un 100% en totes tres mètriques. El bon rendiment d'aquesta etiqueta es deu a la simplicitat de les expressions que cerca, que o bé inclouen el símbol de percentatge (%) o inclouen l'expressió *per cent*.

Respecte a la sisena etiqueta, 'QUANTITY', s'han detectat 8 autèntics positius, cap fals positiu i un fals negatiu. Aquests resultats es tradueixen en un 100% de precisió, un 88,88% de reclam i una mesura F del 94,11%. El fals negatiu es tracta de l'expressió *tres hores*, no detectada per l'expressió regular perquè no inclou numerals expressats en paraules. Tot i això, la baixa quantitat d'entitats nomenades detectades per aquesta etiqueta suggereix que l'avaluació no és significativa. Per obtenir unes dades més fiables, s'hauria d'avaluar un text que contingui moltes més expressions de mesura.

Finalment, de la setena etiqueta, 'CARDINAL', s'han detectat 101 autèntics positius, 12 falsos positius i 29 falsos negatius. Ha obtingut un 89,38% de precisió, un 77,69% de reclam i una mesura F del 83,12%. Els falsos positius són principalment numerals en xifres que no han detectat correctament el substantiu que el seguien o que han detectat el mot que el seguien quan no era un substantiu. En canvi, els falsos negatius són principalment numerals expressats en paraules seguides de substantiu, construccions que l'expressió regular dissenyada per detectar aquesta etiqueta no té en compte.

En conjunt, el rendiment de totes les etiquetes es podria millorar si es consideressin més situacions d'ús real, especialment per a les etiquetes que inclouen expressions no numerals, com és el cas de l'etiqueta 'TIME'. Així mateix, una revisió exhaustiva i metòdica de les expressions regulars permetria millorar tant el rendiment com l'eficàcia del model. Aquesta necessitat es veu clarament exemplificada en la detecció de falsos positius, com és el cas de *onze* per a l'etiqueta 'ORDINAL'.

Amb tot, la immensitat del llenguatge natural limita la capacitat d'acció de les expressions regulars. És per aquesta raó que l'ús de mètodes basats en regles és més efectiu per a textos més especialitzats, en què els patrons lingüístics són més previsibles i menys variables. En canvi, les tècniques d'aprenentatge automàtic demostren una major capacitat per adaptar-se i obtenir millors resultats en textos de naturalesa més diversa, gràcies a la seva habilitat per aprendre i generalitzar a partir de grans volums de dades lingüístiques.

Capítol 6

Aplicació dels resultats en una eina TAO

En aquest capítol s'explorarà la utilitat d'una llista d'entitats nomenades creada a partir d'un text traduïble mitjançant una eina de traducció assistida per ordinador. Per poder treballar amb aquesta llista d'entitats nomenades, s'han escrit un programa senzill que permet traslladar les entitats nomenades al format *.csv*. Aquest document inclou dues columnes, una amb l'entitat nomenada com una cadena de caràcters i una altra amb l'etiqueta corresponent. Al seu torn, aquest format pot ser importat per un programari de fulls de càlcul com Microsoft Excel per treballar amb el contingut més fàcilment. A més, el format nadiu de Microsoft Excel, *.xlsx*, és fàcilment importable com a base de dades terminològica a l'eina TAO MemoQ. És per aquesta raó que s'ha escollit aquesta eina TAO com a programari de treball.

A tall d'exemple, s'ha pres el text utilitzat en el capítol quatre en què es mostrava la detecció de totes les etiquetes alhora. Amb les entitats obtingudes organitzades en columnes en un document de format *.xlsx*, el traductor o traductora pot veure totes les entitats nomenades candidates a formar part d'un glossari creat *ad hoc* per al projecte de traducció. Un cop descartades les menys interessants, el traductor o traductora pot afegir una tercera columna per a l'idioma meta i escriure-hi la traducció. Com es pot observar a la Figura 6.1, en l'exemple proposat s'han descartat les entitats *setena*, *l'any passat*, *2020* i *mensuals*. La resta d'entitats s'han traduït o adaptat per reflectir les convencions de l'anglès dels Estats Units.

Convé destacar que, per a textos molt més llargs, és possible que una mateixa entitat aparegui repetidament. Tot i que aquesta entitat es mostraria diversos cops al full de càlcul creat, el nombre de repeticions és un bon indicatiu per decidir si és adequat incloure l'entitat nomenada al glossari i així mantenir la coherència al llarg del text. En cas que s'hagi decidit incloure l'entitat nomenada en qüestió, es poden

Català	Tipus EN	English
Christine Lagarde	PER	Christine Lagarde
Banc Central Europeu	ORG	European Central Bank
11 d'abril de 2024	DATE	April 11, 2024
Frankfurt	MISC	Frankfurt
4,5%	PERCENT	4.5%
100 €	MONEY	€ 100
1.200 kilòmetres	QUANTITY	750 miles
5.000 persones	CARDINAL	5,000 people
Barcelona	LOC	Barcelona

Figura 6.1: Entitats nomenades que formaran part del glossari *ad hoc* del projecte.

esborrar totes les aparicions a la llista excepte una perquè esdevingui un terme en el glossari del projecte.

Un cop escollides les entitats nomenades que formaran part del glossari i la forma en què es traduiran, ja es pot crear el projecte de MemoQ, incloure el text a traduir, la memòria de traducció del projecte i el glossari en qüestió. En importar el glossari, cal especificar que la primera columna representa el terme en català, la tercera representa l'anglès dels Estats Units —o l'idioma al qual es vulgui traduir— i la segona, en què s'indica el tipus d'etiqueta assignada a la entitat nomenada, es pot incloure com a definició per accedir a aquesta informació fàcilment. La Figura 6.2 mostra una entrada del glossari de la mateixa manera en què apareix a la interfície de MemoQ.

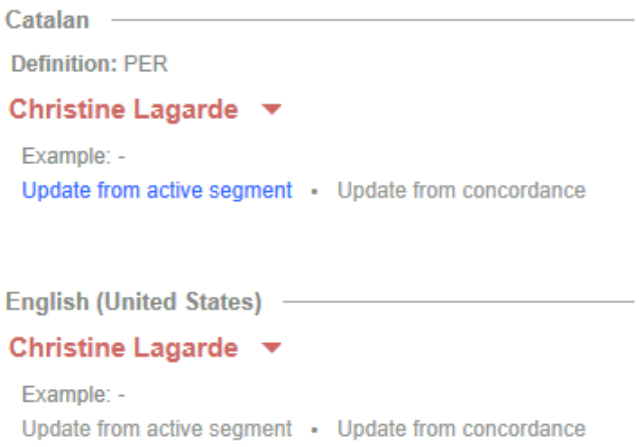

Figura 6.2: Exemple d'un terme detectat en el text a traduir.

Així doncs, l'ús d'aquesta mena de glossaris *ad hoc* per a un text en concret pot ajudar a mantenir la coherència durant el procés de traducció. Tot i que és cert que algunes de les etiquetes com l'etiqueta 'TIME' o l'etiqueta 'ORDINAL' són menys útils que d'altres, les possibles aplicacions de les llistes d'entitats nomenades van més enllà de la creació de glossaris. Dependrà de l'usuari del model de REN implementat trobar més utilitats per a les llistes d'entitats nomenades creades.

Capítol 7

Conclusions

Aquest treball de fi de grau ha tingut com a objectiu el desenvolupament d'un model de reconeixement d'entitats nomenades per a la llengua catalana, combinant tècniques d'aprenentatge automàtic i regles específiques per millorar les capacitats existents dels models de programari lliure. Com a resultat, el model de REN desenvolupat ha satisfet les expectatives inicials a causa del seu bon funcionament i de la diversitat d'entitats nomenades que és capaç de detectar. Això és especialment notable en una situació en què no existeixen altres models plenament operatius amb els quals comparar els resultats.

L'enfocament utilitzat, basat en la distinció per etiquetes de cerca, ha proporcionat una flexibilitat molt convenient al llarg de tot el treball. Pel que fa a la cerca d'entitats nomenades, aquesta flexibilitat ha permès que les expressions regulars tinguessin un propòsit concret que, al seu torn, ha facilitat altres tasques com l'establiment de prioritats entre entitats nomenades.

Quant a l'avaluació, el càlcul dels paràmetres de precisió, reclam i mesura F ha permès analitzar detalladament el rendiment de les diverses parts que conformen el model i identificar les entitats nomenades per a les quals es podria millorar el funcionament. Això obre una nova línia de recerca centrada en la modificació de les expressions regulars i la millora consegüent de rendiment del model desenvolupat. Amb tot, els resultats obtinguts a partir de l'avaluació del model, amb una mesura F de més del 82%, han suposat l'acompliment dels objectius del treball.

En termes de possibles aplicacions, s'ha explorat la creació de glossaris específics per a textos concrets. No obstant això, una altra possible línia de recerca es podria centrar en la definició de noves aplicacions per aprofitar el potencial del model desenvolupat.

D'altra banda, és important destacar que una part significativa de l'esforç invertit en aquest treball de final de grau s'ha centrat en l'elaboració d'un codi que no ha quedat completament reflectit en la

redacció. Pot semblar que els resultats obtinguts han sorgit espontàniament, però sense la feina dedicada al desenvolupament d'aquest codi, els resultats finals haurien sigut molt diferents. Per a aquells interessats en aquesta part més tècnica i menys visible del treball, es recomana revisar tot el codi utilitzat, disponible a l'annex corresponent i al repositori d'aquest treball de fi de grau (Jacas, 2024).

En definitiva, aquest treball ha aconseguit els seus objectius inicials i ha demostrat la viabilitat i l'eficàcia d'un model de REN per al català basat en tècniques híbrides. Els resultats obtinguts no només validen la metodologia utilitzada, sinó que també presenten noves vies de recerca i aplicació en el camp del processament del llenguatge natural, especialment per a llengües amb menys presència a la xarxa com el català.

Bibliografia

Alfred, R., Leong, L. C., On, C. K., & Anthony, P. (2014). Malay named entity recognition based on rule-based approach. *International Journal of Machine Learning and Computing*, *4*(3), 300 - 306. https://doi.org/10.7763/IJMLC.2014.V4.428

Appelt, D. E., Hobbs, J. R., Bear, J., Israel, D., Kameyama, M., Kehler, A., Martin, D., Myers, K., & Tyson, M. (1995). SRI International FASTUS SystemMUC-6 Test Results and Analysis. *Sixth Message Understanding Conference (MUC-6): Proceedings of a Conference Held in Columbia, Maryland, November 6-8, 1995*. Consultat el 18 de abril de 2024, a partir de https://aclanthology.org/M95-1019

Appendix C: Named Entity Task Definition (v2.1). (1995). *Sixth Message Understanding Conference (MUC-6): Proceedings of a Conference Held in Columbia, Maryland, November 6-8, 1995*. Consultat el 11 de març de 2024, a partir de https://aclanthology.org/M95-1024

Carreras, X., Màrquez, L., & Padró, L. (2003). Named entity recognition for catalan using spanish resources. *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics - EACL '03*, *1*, 43. https://doi.org/10.3115/1067807.1067815

Chinchor, N., & Robinson, P. (1998). Appendix E: MUC-7 Named Entity Task Definition (version 3.5). *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29 - May 1, 1998*. Consultat el 19 de abril de 2024, a partir de https://aclanthology.org/M98-1028

Chiticariu, L., Krishnamurthy, R., Li, Y., Reiss, F., & Vaithyanathan, S. (2010, octubre). Domain Adaptation of Rule-Based Annotators for Named-Entity Recognition Tasks. A H. Li & L. Màrquez (Ed.), *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing* (p. 1002 - 1012). Association for Computational Linguistics. Consultat el 11 de març de 2024, a partir de https://aclanthology.org/D10-1098

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, juny). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. A J. Burstein, C. Doran & T. Solorio (Ed.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*

(p. 4171 - 4186). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

Drovo, M. D., Chowdhury, M., Uday, S. I., & Das, A. K. (2019). Named Entity Recognition in Bengali Text Using Merged Hidden Markov Model and Rule Base Approach. *2019 7th International Conference on Smart Computing & Communications (ICSCC)*, 1 - 5. https://doi.org/10.1109/ICSCC.2019.8843661

Grefenstette, G. (1999). Tokenization. A H. van Halteren (Ed.), *Syntactic wordclass tagging* (p. 117 - 133). Springer Netherlands. https://doi.org/10.1007/978-94-015-9273-4_9

Jacas, A. (2024). *GitHub - albertjacas/RENCatala: Codi font i corpus anotat del treball de final de grau d'Albert Jacas Mateu*. Consultat el 27 de maig de 2024, a partir de https://github.com/albertjacas/RENCatala

Jehangir, B., Radhakrishnan, S., & Agarwal, R. (2023). A survey on Named Entity Recognition — datasets, tools, and methodologies. *Natural Language Processing Journal*, *3*, 100017. https://doi.org/10.1016/j.nlp.2023.100017

Jones, K. S. (1994). Natural language processing: A historical review. A A. Zampolli, N. Calzolari & M. Palmer (Ed.), *Current issues in computational linguistics: In honour of don walker* (p. 3 - 16). Springer Netherlands. https://doi.org/10.1007/978-0-585-35958-8_1

Language and machines: Computers in translation and linguistics [Pages: 9547]. (1966, 1 de gener). National Academies Press. https://doi.org/10.17226/9547

Li, J., Sun, A., Han, J., & Li, C. (2022). A Survey on Deep Learning for Named Entity Recognition [Conference Name: IEEE Transactions on Knowledge and Data Engineering]. *IEEE Transactions on Knowledge and Data Engineering*, *34*(1), 50 - 70. https://doi.org/10.1109/TKDE.2020.2981314

Mansouri, A., Affendey, L., & Mamat, A. (2008). Named Entity Recognition Approaches. *Int J Comp Sci Netw Sec*, *8*.

Martinez, A. R. (2012). Part-of-speech tagging [_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/wics.195]. *WIREs Computational Statistics*, *4*(1), 107 - 113. https://doi.org/10.1002/wics.195

Nadeau, D., & Sekine, S. (2007). A survey of named entity recognition and classification [Publisher: John Benjamins]. *Lingvisticæ Investigationes*, *30*(1), 3 - 26. https://doi.org/10.1075/li.30.1.03nad

Rau, L. (1991). Extracting company names from text. *The Seventh IEEE Conference on Artificial Intelligence Application [1991] Proceedings*, *1*, 29 - 32. https://doi.org/10.1109/CAIA.1991.120841

Riaz, K. (2010, juliol). Rule-Based Named Entity Recognition in Urdu. A A. Kumaran & H. Li (Ed.), *Proceedings of the 2010 Named Entities Workshop* (p. 126 - 135). Association for Computational Linguistics. Consultat el 11 de març de 2024, a partir de https://aclanthology.org/W10-2419

Strötgen, J., & Gertz, M. (2010, juliol). HeidelTime: High Quality Rule-Based Extraction and Normalization of Temporal Expressions. A K. Erk & C. Strapparava (Ed.), *Proceedings of the 5th*

International Workshop on Semantic Evaluation (p. 321 - 324). Association for Computational Linguistics. Consultat el 19 de abril de 2024, a partir de https://aclanthology.org/S10-1071

Taule, M., Marti, M. A., & Recasens, M. (2008). AnCora: Multilevel annotated corpora for catalan and spanish.

Tjong Kim Sang, E. F. (2002). Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition. *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*. Consultat el 11 de març de 2024, a partir de https://aclanthology.org/W02-2024

Annex: Codi utilitzat

Bloc de codi que inclou la definició de funcions que s'utilitzaran per a altres blocs de codi:

```
import re
from spacy.tokens import Span

# Una clase per definir temporalment entitats nomenades.
class EntitatNomenada:

    # Constructor de la classe.
    def __init__(self, etiqueta, inici, longitud, tokens):

        self.etiqueta = etiqueta
        self.inici = inici
        self.longitud = longitud
        self.fi = inici + longitud
        self.tokens = tokens

    # Exporta l'EntitatNomenada en el format de spaCy.
    def a_spacy(self, doc):

        return Span(doc, self.inici, self.fi, self.etiqueta)

    # Exporta l'EntitatNomenada com una tupla.
    def a_tupla(self):

        return (self.etiqueta, self.inici, self.longitud, tuple(self.tokens))

    # Imprimeix els valors per poder veure'ls fàcilment.
```

```
    def print(self):

        print("{} ({}-{}) =".format(self.etiqueta, self.inici, self.fi, self.tokens),
            self.tokens)

    # Verifica que aquesta entitat nomenada és correcta.
    def verificar(self, tokens_reals):

        tokens_extrets = tokens_reals[self.inici:self.fi]
        tokens_strip = [t.strip(',.') for t in tokens_extrets]
        assert(self.tokens == tokens_extrets or self.tokens == tokens_strip)


# Afegir una nova entitat nomenada al document d'spaCy.
def afegir_entitat_nomenada(doc, entitat_nomenada):

    # Obtenim la llista d'entitats com una llista de Python.
    llista_entitats = list(doc.ents)

    # Afegim la nostra llista en el format de spaCy.
    llista_entitats.append(entitat_nomenada.a_spacy(doc))

    # Estamblim aquesta llista com a llista d'entitats al document.
    doc.set_ents(llista_entitats)

    return doc

# Troba tots els fragment del text que coincideixin amb l'expressió
# regular i els retorna com una llista.
def obtenir_totes_les_coincidencies(regex, text):

    return [match[0] for match in re.finditer(regex, text)]

# Converteix una serie de troballes en un text en referencies al text
# tokenitzat utilitzant els indexs dels tokens corresponents.
def convertir_fragments_a_referencies(tokens, fragments_tokenitzats, etiqueta):
```

```
    # Inicialitza la llista que retornarem.
    llista_referencies = []

    # Inicialitza un minim on cercar.
    # Això ens garanteix que el codi que apliquem s'efectua sobre la següent
    # coincidencia amb els tokens i no sempre sobre la primera.
    index_min_cerca = 0

    # Per cada fragment trobat, itera.
    for fragment in fragments_tokenitzats:

        # Troba l'index on comença la subllista dins la llista de fragments.
        index_trobat = troba_llista_dins_de_llista(fragment, tokens, index_min_cerca)
        index_min_cerca = index_trobat + len(fragment)

        # Si no podem trobar el fragment en el conjunt de tokens, significa que és
        # un error i hem de descartar aquest fragment.
        if index_trobat < 0:
            print('DESCARTAT:', fragment)
            continue

        # Creem una nova entitat nomenada i l'incorporem a la llista.
        llista_referencies.append(EntitatNomenada(etiqueta, index_trobat, len(fragment),
            fragment))

    return llista_referencies

# Troba el primer index en què apareix una llista donada dins d'una altra.
def troba_llista_dins_de_llista(subllista, llista, index_minim=0):

    # Calcula la llargada de la llista que busquem.
    longitut_cadena = len(subllista)

    # Iterem per les possibles subllistes d'aquesta durada.
    for index in range(index_minim, len(llista) - longitut_cadena + 1):

        subllista_actual = llista[index:index + longitut_cadena]
```

```
        # Probem també a fer un trimming del text per arreglar problemes
        # del tokenitzador.
        sublista_actual_trim = [t.strip(',.') for t in sublista_actual]

        # Si aquesta sublista es la mateixa que busquem, retorna
        # el seu índex i deixem de buscar.
        if sublista_actual == sublista or sublista_actual_trim == sublista:
            return index

    return -1

# Retorna si dues entitats nomenades se solapen
def comprova_si_solapa(entitat_1, entitat_2):

    return (entitat_1.inici <= entitat_2.inici and entitat_1.fi - 1 >= entitat_2.inici)
        or \
        (entitat_1.inici <= entitat_2.fi - 1 and entitat_1.fi - 1 >= entitat_2.fi - 1)

# Verifica que les dues entitats nomenades són equivalents.
def entitats_nomenades_cmp(entitat_1, entitat_2):

    return entitat_1.etiqueta == entitat_2.etiqueta and \
        entitat_1.inici == entitat_2.inici and \
        entitat_1.longitud == entitat_2.longitud and \
        entitat_1.fi == entitat_2.fi and \
        entitat_1.tokens == entitat_2.tokens
```

Bloc de codi dedicat a la cerca d'entitats nomenades:

```
SAVE_IMAGES = False

import re, spacy, pickle

from utils import EntitatNomenada, obtenir_totes_les_coincidencies,
    convertir_fragments_a_referencies, afegir_entitat_nomenada, comprova_si_solapa,
```

```
    entitats_nomenades_cmp
from spacy import displacy
from pathlib import Path

# Inicialitzem spacy en català.
nlp = spacy.load("ca_core_news_sm")

# Text on buscarem entitats nomenades.
with open('corpus_raw.txt', 'rb') as file:
    text_sencer = file.read().decode('utf-8')

# Tokenitza el text.
doc_spacy = nlp(text_sencer)

# Obté els tokens com strings.
tokens = [token.text for token in doc_spacy]

# Inicialitzem una llista buida on anirem guardant les entitats
# nomenades resultants.
entitats_nomenades = []


# Definim l'expressió regular per trobar dates senceres i hores
regex_dates_hores = (r''
    r'\b(?:0?[1-9]|[12][0-9]|3[01])[-/](?:0?[1-9]|1[0-2])'
        r'([-/](\d{2}|\d{4}))?\b|'
    r'\b(?:0?[1-9]|[12][0-9]|3[01])\s(de\s|d\')(?:gener|febrer|març|abril|maig|juny|'
        r'juliol|agost|setembre|octubre|novembre|desembre)(\s(del?\s)(\d{2,4}))?\b|'
    r'\b(([Ee]l\s)?gener|([Ee]l\s)?febrer|([Ee]l\s)?març|([Ll]\')?abril|([Ee]l\s)?maig'
        r'|([Ee]l\s)?juny|([Ee]l\s)?juliol|([Ll]\')?agost|([Ee]l\s)?setembre|([Ll]\')?'
        r'octubre|([Ee]l\s)?novembre|([Ee]l\s)?desembre)(\s(del?\s)(\d{2,4}))\b|'
    r'\b([01]?\d|2[0-3])([:\.][0-5]\d\s?)(hores|h)?\b|'
    r'\b(([Ee]l\s)|[Ll]\'any\s)?\d{4}\b|'
    r'\b[Ss]egle\sM{0,3}(CM|CD|D?C{0,3})(XC|XL|L?X{0,3})(IX|IV|V?I{0,3})\b|'
    r'\d{1,2}\-[GFMAJSOND]')

# Per obtenir les strings completes de cada data trobada
```

```
dates_senceres_hores = obtenir_totes_les_coincidencies(regex_dates_hores, text_sencer)

# Imprimim les dates trobades
print("Dates i hores trobades:", "\n".join(dates_senceres_hores), sep="\n\n")

# Tokenitza les troballes.
# Converteix això en una llista de llistes de tokens en format string.
troballes_tokenitzades = [[t.text for t in nlp(trob)] for trob in dates_senceres_hores]

# Obtenir les referències com a instàncies d'EntitatNomenada.
entitats_nomenades += convertir_fragments_a_referencies(tokens, troballes_tokenitzades,
    'DATE')

# Ara verifiquem si aquestes es corresponen correctament als tokens inicials.
for en in entitats_nomenades:
    en.verificar(tokens)
else:
    print('Tot verificat!')


# Definim l'expressió regular per trobar moments i dates relatius al dia actual
regex_dates_relatives = (r''
    r'\b((?:[Aa]quest\s)?([Dd]illuns|[Dd]imarts|[Dd]imecres|[Dd]ijous|'
        r'[Dd]ivendres|[Dd]issabte|[Dd]iumenge)(?:\sque\sve|\spassat)?)\b|'
    r'\b([Aa]questa\ssetmana|[Aa]quest\smes|[Aa]quest\sany)\b|'
    r'\b(([Ll]a\ssetmana\s|[Ee]l\smes\s|[Ll]\'any\s)(que\sve|passa(t|da)|'
        r'precedent|següent|anterior|posterior))\b|'
    r'\b([Ff]a\s|[Dd]\'aquí\sa?\s?|[Cc]ada\s?)?([Gg]airebé\s|[Tt]ot\sjust\s|'
        r'([Pp]oc\s)?més\s(d\'|de))?([Uu]na?e?s?\s|[Dd](o|ue)s\s|[Tt]res\s|'
        r'[Qq]uatre\s|[Cc]inc\s|[Ss]is\s)(di(a|es)|setman(a|es)|meso?s?|anys?)\b|'
    r'\b([Aa]vui|[Dd]emà|([Aa]bans-d\')?[Aa]hir|[Dd]emà\spassat)\b|'
    r'\b((?:[Jj]ust\s)?[Aa]ra|(?:[Mm]olt\s|[Pp]oc\s|[Jj]ust\s)?[Aa]bans|'
        r'(?:[Pp]oc\s|[Jj]ust\s)?[Dd]esprés)\b|'
    r'\b[AaEe]\s?l\'?\s(mes\s(de\s|d\'))?(gener|febrer|març|abril|maig|juny|juliol|'
        r'agost|setembre|octubre|novembre|desembre)\b|'
    r'\b(((?:[Pp]rimera\s|[Úú]ltima\s)hora\s(?:del\s|de\sla\s))?|([Aa]quest\s|'
        r'[AaEe]l\s|[Ll]a\s|[Aa]\sla\s)?)(matí|migdia|tarda|vespre|nit|matinada)|'
```

```
    r'\b([Ll]a|[Ll]es)\s(una|dues|tres|quatre|cinc|sis|set|vuit|nou|deu|onze|dotze|'
        r'\d{1,2}))(?:\s(del\smatí|del\smigdia|de\sla\starda|del\svespre|de\sla\snit|'
        r'de\sla\smatinada))\b')

# Per obtenir les strings completes de cada data trobada
dates_relatives = obtenir_totes_les_coincidencies(regex_dates_relatives, text_sencer)

# Imprimim les EN trobades
print("Dates relatives trobades:", "\n".join(dates_relatives), sep="\n\n")

# Tokenitza les troballes.
# Converteix això en una llista de llistes de tokens en format string.
troballes_tokenitzades = [[t.text for t in nlp(trob)] for trob in dates_relatives]

# Obtenir les referències com a instàncies d'EntitatNomenada.
entitats_nomenades += convertir_fragments_a_referencies(tokens, troballes_tokenitzades,
    'TIME')

# Ara verifiquem si aquestes es corresponen correctament als tokens inicials.
for en in entitats_nomenades:
    en.verificar(tokens)
else:
    print('Tot verificat!')


# Definim l'expressió regular per trobar freqüències
regex_frequencies = (r''
    r'\b(([Cc]ada|([Uu]na?|[Dd]os|[Dd]ues|[Tt]res|[Qq]uatre|[Cc]inc|[Ss]is|'
        r'[Ss]et|[Vv]uit|[Nn]ou|[Dd]eu))\s(cops?|vegad(a|es)\s(al\s|a\sla\s|'
        r'(a\s)?l\'|per\s))(dia|setmana|mes|any|estiu|tardor|hivern|'
        r'primavera|dilluns|dimarts|dimecres|dijous|divendres|dissabte|diumenge)\b|'
    r'\b[Ee]ls\s(dilluns|dimarts|dimecres|dijous|divendres|dissabte|diumenge)\b|'
    r'\b([Dd]i[aà]ria?|[Ss]etmanal|[Qq]uinzenal|[Pp]eriòdica?|'
        r'[Ff]reqüent|[Oo]casional|([Bb]i|[Tt]ri|[Qq]uadri|[Ss]e)?([Mm]ensuals?|'
        r'mestral|[Aa]nuals?|ennal))(ment)?)\b|'
    r'\b[Ss]empre|[Mm]ai|[Ss]ovint|(([Aa]|[Dd]e)\svegades)|[Dd]e\stant\sen\stant\b')
```

```
# Per obtenir les strings completes de cada freqüència trobada
frequencies = obtenir_totes_les_coincidencies(regex_frequencies, text_sencer)

# Imprimim les EN trobades
print("Expressions de freqüència trobades:", "\n".join(frequencies), sep="\n\n")

# Tokenitza les troballes.
# Converteix això en una llista de llistes de tokens en format string.
troballes_tokenitzades = [[t.text for t in nlp(trob)] for trob in frequencies]

# Obtenir les referències com a instàncies d'EntitatNomenada.
entitats_nomenades += convertir_fragments_a_referencies(tokens, troballes_tokenitzades,
    'TIME')

# Ara verifiquem si aquestes es corresponen correctament als tokens inicials.
for en in entitats_nomenades:
    en.verificar(tokens)
else:
    print('Tot verificat!')

# Definim l'expressió regular per detectar nombres ordinals
regex_ordinals = (r''
    r'\b([Pp]rimer(a|e?s)?|[Ss]egon(a|es)?|[Tt]ercer(a|e?s)?|[Qq]uart(a|e?s)?|'
        r'[Cc]inqu[èe](ns|na|nes)?|[Ss]is[èe](ns|na|nes)?|([Dd]is)?[Ss]et[èe]'
        r'(ns|na|nes)?|([Dd]i)?[Vv]uit[èe](ns|na|nes)?|([Dd]i)?[Nn]ov[èe]'
        r'(ns|na|nes)?|[Dd]es[èe](ns|na|nes)?|[Oo]nz[èe](ns|na|nes)?|[Dd]otz[èe]'
        r'(ns|na|nes)?|[Tt]retz[èe](ns|na|nes)?|[Cc]atorz[èe](ns|na|nes)?|'
        r'[Qq]uinz[èe](ns|na|nes)?|[Ss]etz[èe](ns|na|nes)|[Vv]int[èe](ns|na|nes)??)\b|'
    r'\b\d+(rs?|ts?|è|a|ns?|es)\.?\b|'
    r'\b(?=[MDCLXVI])M{0,3}(CM|CD|D?C{0,3})(XC|XL|L?X{0,3})(IX|IV|V?I{0,3})'
        r'\b(?=\s[A-Z])')

# Per obtenir les strings completes de cada ordinal trobat
ordinals = obtenir_totes_les_coincidencies(regex_ordinals, text_sencer)

# Imprimim les EN trobades
print("Ordinals trobats:", "\n".join(ordinals), sep="\n\n")
```

```
# Tokenitza les troballes.
# Converteix això en una llista de llistes de tokens en format string.
troballes_tokenitzades = [[t.text for t in nlp(trob)] for trob in ordinals]

# Obtenir les referències com a instàncies d'EntitatNomenada.
entitats_nomenades += convertir_fragments_a_referencies(tokens, troballes_tokenitzades,
    'ORDINAL')

# Ara verifiquem si aquestes es corresponen correctament als tokens inicials.
for en in entitats_nomenades:
    en.verificar(tokens)
else:
    print('Tot verificat!')


# Definim l'expressió regular per detectar monedes
regex_monedes = (r''
    r'(\d{1,3}(?:,\d{1,3})?\s?(\.\d{3})*(?:,\d{1,3})?\s?)(mil(ers)?\s(de\s)?)'
        r'?([bm]ilions\s(de\s)?)?((cèntims?|centaus?|¢|penics?)|(afganis?|AFN)'
        r'|(ariarys?|MGA)|(฿|bahts?|THB)|(balboas?|PAB)|(birrs?|ETB)|'
        r'(bolívars?|VEF)|(bolivianos?|BOB)|(cedis?|GHS)|(₵|colons?|CRC)|'
        r'(córdob(a|es)|NIO)|(coron(a|es)|DKK|SKK|EEK|ISK|NOK|SEK|CZK)|'
        r'(dalasis?|GMD)|(denars?|MKD)|(dinars?|DZD|BHD|IQD|JOD|KWD|LYD|TND|'
        r'RSD|IRR)|(dírhams?|AED|MAD)|(dobr(a|es)|STD)|(\$|\$CAN|dòlars?'
        r'|AUD|BSD|BBD|BZD|BMD|BND|KYD|CAD|XCD|USD|FJD|GYD|HKD|JMD|LRD|NAD|NZD'
        r'|SBD|SGD|SRD|TWD|TTD|ZWL)|(₫|dongs?|VND)|(DR|dracm(a|es))|(drams?|AMD)|'
        r'(escuts?|CVE)|(M?€|d?\'?euros?|M?EUR)|(f|FL|flor(í|ins)|ANF|AWG)|'
        r'(fòrints?|HUF)|(FB|FLUX|FS|francs?|BIF|XAF|XOF|XPF|KMF|CDF|DJF|GNF|'
        r'RWF|CHF)|(gourdes?|HTG)|(guaranís?|PYG)|(hrívni(a|es)|UAH)|'
        r'(¥|iens?|iuans?|renminbis?|JPY|CNY)|(kin(a|es)|PGK)|(kips?|LAK)|'
        r'(kun(a|es)|HRK)|(kwach(a|es)|MWK|ZMK)|(kwanz(a|es)|AOA)|(kyats?|MMK)|'
        r'(laris?|GEL)|(lats|LVL)|(leks?|ALL)|(L|lempir(a|es)|HNL)|(leones?|SLL)|'
        r'(leus?|MDL|RON)|(levs?|BGN)|(lilangenis?|SZL)|([££]|lir(a|es)|TRY)|'
        r'(litas|LTL)|([££]|£EG|£IR|£LIB|£SYR|£TQ|(lliur(a|es)(\sesterlin(a|es))?'
        r'|EGP|GBP|GIP|GGP|JEP|LBP|FKP|IMP|SHP|SYP|SDG|SSP))|(lotis?|LSL)|'
        r'(manats?|AZN|TMT)|(DM|MF|marcs?|BAM)|(meticals?|MZN)|(₦|nair(a|es)|NGN)|'
```

```
        r'(nafk(a|es)|ERN)|(ngultrums?|BTN)|(ouguiy(a|es)|MRO)|(pa\'ang(a|ues)|TOP)|'
        r'(pata(ca|ques)|MOP)|(₱|$MEX|$CH|CU|pesos?|ARS|COP|CUC|CUP|DOP|PHP|MXN|'
        r'UYU|CLP)|(pesset(a|es)|pt(a|es)\.?|ESP|M?PTA)|(pul(a|es)|BWP)|'
        r'(quetzals?|GTQ)|(rands?|ZAR)|(R$|reals?|BRL)|(rials?|YER|IRR|OMR)|'
        r'(riels?|KHR)|(ringgits?|MYR)|(riyals?|QAR|SAR)|(rubles?|BYR|RUB)|'
        r'(rupi(a|es)|INR|IDR|MVR|MUR|NPR|PKR|SCR|LKR)|(sols?(?=\s)|PEN)|'
        r'(soms?|KGS|UZS)|(somonis?|TJS)|(sucres?)|(tak(a|es)|BDT)|(tal(a|es)|WST)|'
        r'(tengu?es?|KZT)|(tögrögs?|MNT)|(tolars?)|(vatus?|VUV)|(₩|wons?|KPW|KRW)|'
        r'(£IS|(nous?\s)?xéquels?|ILS)|(xílings?|KES|SOS|TZS|UGX)|(zaires?)|'
        r'(ZL|z[␣ll]otys?|PLN)|¤)')

# Per obtenir les strings completes de cada moneda trobada
monedes = obtenir_totes_les_coincidencies(regex_monedes, text_sencer)

# Imprimim les EN trobades
print("Monedes trobades:", "\n".join(monedes), sep="\n\n")

# Tokenitza les troballes.
# Converteix això en una llista de llistes de tokens en format string.
troballes_tokenitzades = [[t.text for t in nlp(trob)] for trob in monedes]

# Obtenir les referències com a instàncies d'EntitatNomenada.
entitats_nomenades += convertir_fragments_a_referencies(tokens, troballes_tokenitzades,
    'MONEY')

# Ara verifiquem si aquestes es corresponen correctament als tokens inicials.
for en in entitats_nomenades:
    en.verificar(tokens)
else:
    print('Tot verificat!')


# Expressió regular per detectar percentatges
regex_percentatges = (r''
    r'\-?(?:\d{1,3}(?:\.\d{3})*|\d+)(?:,\d+)?\s?(%|per\scent)|'
    r'\b(?:zero|u|dos|tres|quatre|cinc|sis|set|vuit|nou|deu|onze|dotze|tretze|'
        r'catorze|quinze|setze|disset|divuit|dinou|vint|trenta|quaranta|'
```

```
        r'cinquanta|seixanta|setanta|vuitanta|noranta|cent)\sper\scent\b')

# Per obtenir les strings completes de cada percentatge trobat
percentatges = obtenir_totes_les_coincidencies(regex_percentatges, text_sencer)

# Imprimim les EN trobades
print("Mesures trobades:", "\n".join(percentatges), sep="\n\n")

# Tokenitza les troballes.
# Converteix això en una llista de llistes de tokens en format string.
troballes_tokenitzades = [[t.text for t in nlp(trob)] for trob in percentatges]

# Obtenir les referències com a instàncies d'EntitatNomenada.
entitats_nomenades += convertir_fragments_a_referencies(tokens, troballes_tokenitzades,
    'PERCENT')

# Ara verifiquem si aquestes es corresponen correctament als tokens inicials.
for en in entitats_nomenades:
    en.verificar(tokens)
else:
    print('Tot verificat!')


# Expressió regular per detectar unitats de mesura
regex_mesures = (r''
    r'\b((-?\d{1,3}(?:[.,]\d{3})*(?:,\d+)?\s*))\s?
    r'((([QRYZEPTGMKkhdcmµnpfzyrq]|da)?'
        r'((m|g|s|A|K|mol|cd|N|Pa|J|W|C|V|S|F|T|Wb|H|rad|sr|Hz|lm|lx|Bq|Gy|Sv|'
        r'kat|B|b|L|l|cal|fg|mi|UA|ly|pc|t|HP|hp|ac|Ac|[ºª]C|[ºª]F|P|bar|atm)¯?'
        r'[¹²³123]?([p\.\/]?([QRYZEPTGMKkhdcmµnpfazyrq]?'
        r'(m|g|s|A|K|mol|cd|h)¯?[¹²³123]?))*))|
    r'\b([Qq]uett[aà]|[Rr]onn[aà]|[Yy]ott[aà]|[Zz]ett[aà]|[Ee]x[aà]|[Pp]et[aà]|'
        r'[Tt]er[aà]|[Gg]ig[aà]|[Mm]eg[aà]|[KkQq]u?il[oò]|[Hh]ect[oò]?|[Dd]ec[aà]|'
        r'[Dd]ec[ií]|[Cc]ent[ií]|[Mm]il·l[ií]|[Mm]icr[oò]|[Nn]an[oò]|[Pp]ic[oò]|'
        r'[Ff]emt[oò]|[Aa]tt[oò]|[Zz]ept[oò]|[Yy]oct[oò]|[Rr]ont[oò]|[Qq]ect[oò])?'
        r'(metres?|grams?|segons?|minuts?|hor(a|es)|amperes?|(graus?\s)kelvin|'
        r'candel(a|es)|mols?|newtons?|pascals?|joules?|watts?|coulombs?|volts?|'
```

```
        r'ohms?|siemens?|farads?|tesl(a|es)|webers?|henrys?|radiant?s?|'
        r'estereoradiant?s?|hertzs?|lumens?|luxs?|becquerels?|grays?|sieverts?|'
        r'katals?|bels?|bytes?|bits?|litres?|gal[óo](ns)?|calori(a|es)|'
        r'frigori(a|es)|mill(a|es)(\snàuti(ca|ques))?|llegu(a|es)|iard(a|es)|'
        r'peus?|polzad(a|es)|bra(ça|ces)|unitats?\sastronòmi(ca|ques)|'
        r'anys?\sllum|pàrsecs?|ton(a|es)|tonelad(a|es)|un(ça|ces)|cavalls?|'
        r'àre(a|es)|acres?|graus?|graus?\s(Celsius|centígrads?'
        r'|Fahrenheits?)|poises?|bars?|atmosfer(a|es)|psis?)(\sper)?\s?'
        r'(metres?|grams?|segons?|amperes?|graus?\skelvin|candel(a|es)|mols?|'
        r'hor(a|es))?((al)?\squadrats?|al\scub|cúbics?)?)\b')

# Per obtenir les strings completes de cada mesura trobada
mesures = obtenir_totes_les_coincidencies(regex_mesures, text_sencer)

# Imprimim les EN trobades
print("Mesures trobades:", "\n".join(mesures), sep="\n\n")

# Tokenitza les troballes.
# Converteix això en una llista de llistes de tokens en format string.
troballes_tokenitzades = [[t.text for t in nlp(trob)] for trob in mesures]

# Obtenir les referències com a instàncies d'EntitatNomenada.
entitats_nomenades += convertir_fragments_a_referencies(tokens, troballes_tokenitzades,
    'QUANTITY')

# Ara verifiquem si aquestes es corresponen correctament als tokens inicials.
for en in entitats_nomenades:
    en.verificar(tokens)
else:
    print('Tot verificat!')


# Expressió regular per detectar quantitats
regex_quantitats = r''
    r'(\d+(?:[.,]\d{3})*(?:,\d+)?\s*)((?!(?:a|amb|en|entre|per|sense|'
        r'sota|sobre|del?|pel|al|i|o|que)\s)[^\s,\.\)\%\;]+)(?=\s|,|\.)'
```

```
# Per obtenir les strings completes de cada quantitat trobada
quantitats = obtenir_totes_les_coincidencies(regex_quantitats, text_sencer)

# Imprimim les EN trobades
print("Quantitats trobades:", "\n".join(quantitats), sep="\n\n")

# Tokenitza les troballes.
# Converteix això en una llista de llistes de tokens en format string.
troballes_tokenitzades = [[t.text for t in nlp(trob)] for trob in quantitats]

# Obtenir les referències com a instàncies d'EntitatNomenada.
entitats_nomenades += convertir_fragments_a_referencies(tokens, troballes_tokenitzades,
    'CARDINAL')

# Ara verifiquem si aquestes es corresponen correctament als tokens inicials.
for en in entitats_nomenades:
    en.verificar(tokens)
else:
    print('Tot verificat!')


# Ordenem les prioritats de cada etiqueta per sobreescriure les EN que apareixen
# en més d'una RegEx
prioritats = {
    'PERCENT' :     7,
    'MONEY' :       6,
    'DATE' :        5,
    'QUANTITY' :    4,
    'ORDINAL' :     3,
    'TIME' :        2,
    'CARDINAL' :    1,
    'ERROR' :       0,
    'PERCENT-ERROR' :     0,
    'ORDINAL-ERROR' :     0,
    'MONEY-ERROR' :       0,
    'DATE-ERROR' :        0,
    'QUANTITY-ERROR' :    0,
```

```
    'TIME-ERROR' :        0,
    'CARDINAL-ERROR' :    0,
    'ERROR-ERROR' :       0
}

# Bucle en què iterem per totes les combinacions d'etiquetes
for i, en1 in enumerate(entitats_nomenades):
    for j, en2 in enumerate(entitats_nomenades):

        # Condicional per evitar comparacions entre una mateixa etiqueta
        if i == j:
            continue

        # Condicional per evitar repetir comparacions que ja hem fet
        if i > j:
            continue

        # Condicional per assignar la categoria ERROR si etiquetes se solapen
        if comprova_si_solapa(en1, en2):
            if prioritats[en1.etiqueta] <= prioritats[en2.etiqueta]:
                if '-ERROR' not in en1.etiqueta:
                    en1.etiqueta += '-ERROR'
            else:
                if '-ERROR' not in en2.etiqueta:
                    en2.etiqueta += '-ERROR'

# Iterem per les entitats en sentit contrari per eliminar els errors
# Ho fem en sentit contrari per evitar errors pel desplaçament dels índexs
for i in reversed(range(len(entitats_nomenades))):

    # Elimina els que han estat marcats com errors.
    if '-ERROR' in entitats_nomenades[i].etiqueta:
        del entitats_nomenades[i]

for en in entitats_nomenades:
    en.print()
```

```
# Convertim totes les EN trobades per Spacy al nostre format.
entitats_spacy = [EntitatNomenada(e.label_, e.start, e.end - e.start,
    tokens[e.start:e.end]) for e in doc_spacy.ents]

DESCARTAR_REGEX = True

# Bucle en què iterem per totes les combinacions d'etiquetes nostres i de spaCy
for i, en_spacy in enumerate(entitats_spacy):
    for j, en_regex in enumerate(entitats_nomenades):

        # Condicional per assignar la categoria ERROR si etiquetes se solapen
        if comprova_si_solapa(en_spacy, en_regex):

            if DESCARTAR_REGEX:

                en_regex.etiqueta += '-ERROR'

            else:

                en_spacy.etiqueta += '-ERROR'

# L'afegim a la llista general d'EN
for i, en_spacy in enumerate(entitats_spacy):
    entitats_nomenades.append(en_spacy)

# Iterem per les entitats en sentit contrari per eliminar els errors
# Ho fem en sentit contrari per evitar errors pel desplaçament dels índexs
for i in reversed(range(len(entitats_nomenades))):

    # Elimina els que han estat marcats com errors.
    if '-ERROR' in entitats_nomenades[i].etiqueta:
        del entitats_nomenades[i]

# Eliminem totes les EN perquè les hem inclòs a la nostra llista
doc_spacy.ents = []

# Afegim totes les EN que hem trobat i/o no hem descartat
```

```
for entitat in entitats_nomenades:
    afegir_entitat_nomenada(doc_spacy, entitat)

# Guardem les entitats nomenades en un arxiu Pickle
with open('entitats_nomenades.corpus_raw.pickle', 'wb') as file:
    pickle.dump(entitats_nomenades, file)

# Guardem els tokens en un arxiu Pickle
with open('tokens.corpus_raw.pickle', 'wb') as file:
    pickle.dump(tokens, file)

# Bloc opcional per mostrar en pantalla les EN detectades al text

html = displacy.render(doc_spacy, style="ent", jupyter=not SAVE_IMAGES)

if SAVE_IMAGES:
    output_path = Path("C:\\Users\\alber\\OneDrive\\Documents\\Cosetes\\prova_imatge1
        .html")
    output_path.open("w", encoding="utf-8").write(html)
```

Bloc de codi dedicat a l'obtenció de les mètriques d'avaluació:

```
import conllu, pickle, csv
from itertools import zip_longest
from conllu import parse
from io import open

from utils import EntitatNomenada

# Llegim les variables resultants d'executar les regex
with open('entitats_nomenades.corpus_raw.pickle', 'rb') as file:
    entitats_nomenades_res = pickle.load(file)

with open('tokens.corpus_raw.pickle', 'rb') as file:
    tokens_res = pickle.load(file)
```

```
# Guardem el corpus de test anotat manualment a la variable dades
with open("test_docs_anotat.conllu", "r", encoding="utf-8") as dades:

    # Llegeix els continguts del fitxer CONLLU i els guarda a la variable anotacions
    anotacions = dades.read()

# Segmentem en frases tot el corpus i les guardem a la variable frases
frases = conllu.parse(anotacions)

# Obtenim els tokens a partir de l'arxiu de test
tokens_test = []
for frase in frases:
    for token in frase:
        tokens_test.append((token))

# Comparem els dos conjunts de tokens per garantir que les dades son consistents
if tokens_res == list(map(str, tokens_test)):
    print("Tot perfecte.")

# Retorna l'etiqueta de la EN i el tipus ('B', 'I', 'O').
def obtenir_tipus_entitat(token):

    # Si no té dades, retornem buit.
    if token['misc'] is None:
        return 'O', None

    # Iterem per les claus del diccionari de dades.
    for clau in token['misc']:

        if clau[0:2] == 'B-':
            return 'B', clau[2:]

        if clau[0:2] == 'I-':
            return 'I', clau[2:]

    return 'O', None
```

```
# Iterem per tots els tokens per cercar EN
i = 0
entitats_nomenades_test = []
while i < len(tokens_test):

    # Obtenim el token corresponent
    token_i = tokens_test[i]
    estat, etiqueta = obtenir_tipus_entitat(token_i)
    inici = i

    # Ja hem accedit al token, passem al següent ja en cas d'entrar
    # al bloc condicional
    i += 1

    # Si marca l'inici d'una EN, entrem al bloc
    if estat == 'B':

        # Obtenim el token corresponent.
        token_i = tokens_test[i]
        estat, _ = obtenir_tipus_entitat(token_i)
        final = i

        # Mentre la EN continuï o no s'hagi acabat el document
        while estat == 'I' and i < len(tokens_test):

            # Saltem al token següent
            i += 1

            # Actualitzem la informació
            token_i = tokens_test[i]
            estat, _ = obtenir_tipus_entitat(token_i)
            final = i

        # Creem la instància d'EntitatNomenada i l'afegim a l'estructura
        entitats_nomenades_test.append(EntitatNomenada(etiqueta,
            inici, final - inici, tokens_res[inici:final]))
```

```
# Ordenem les llistes pel seu índex de token
entitats_nomenades_res.sort(key=lambda x: x.inici)
entitats_nomenades_test.sort(key=lambda x: x.inici)

for en1, en2 in zip_longest(entitats_nomenades_res, entitats_nomenades_test):

    print("RES:", end=" ")
    if en1 is None:
        print("-")
    else:
        en1.print()

    print("TEST:", end=" ")
    if en2 is None:
        print("-")
    else:
        en2.print()

# Transformem les instàncies d'EntitatNomenada a tuples per poder operar-hi fàcilment
entitats_nomenades_res_tuples = set(map(lambda x: x.a_tupla(),
    entitats_nomenades_res))
entitats_nomenades_test_tuples = set(map(lambda x: x.a_tupla(),
    entitats_nomenades_test))

# Operem amb els conjunts per obtenir la intersecció i subtraccions
true_positives = entitats_nomenades_res_tuples & entitats_nomenades_test_tuples
false_positives = entitats_nomenades_res_tuples - entitats_nomenades_test_tuples
false_negatives = entitats_nomenades_test_tuples - entitats_nomenades_res_tuples

# Obtenim les mètriques de cada categoria
metrica_tp = len(true_positives)
metrica_fp = len(false_positives)
metrica_fn = len(false_negatives)

# Obtenim les mètriques complexes
precisio = metrica_tp / (metrica_tp + metrica_fp)
reclam = metrica_tp / (metrica_tp + metrica_fn)
```

```
f1 = 2 * precisio * reclam / (precisio + reclam)

print("Autèntics positius:", metrica_tp)
print("Falsos positius:", metrica_fp)
print("Falsos negatius:", metrica_fn)
print("==========")
print("Precisió: {}%".format(int(precisio * 10000) / 100))
print("Reclam: {}%".format(int(reclam * 10000) / 100))
print("Mesura F: {}%".format(int(f1 * 10000) / 100))

# Imprimim els falsos negatius
for en in false_negatives:
    print(en)

# Imprimim els falsos positius
for en in false_positives:
    print(en)

# Obtenim la llista d'etiquetes
etiquetes_disponibles = set(map(lambda e: e[0], entitats_nomenades_test_tuples))
etiquetes_spacy_org = {'LOC', 'MISC', 'ORG', 'PER'}
etiquetes_propies = set(e for e in etiquetes_disponibles if e not in
    etiquetes_spacy_org)

# Obtenim les mètriques pel sistema original de spacy
en_resultat_spacy = set(filter(lambda e: e[0] in etiquetes_spacy_org, entitats_nomenades_res_tuples)
en_test_spacy = entitats_nomenades_test_tuples

# Operem amb els conjunts per obtenir la intersecció i subtraccions.
true_positives_spacy = en_resultat_spacy & en_test_spacy
false_positives_spacy = en_resultat_spacy - en_test_spacy
false_negatives_spacy = en_test_spacy - en_resultat_spacy

# Obtenim les mètriques de cada categoria
metrica_tp = len(true_positives_spacy)
metrica_fp = len(false_positives_spacy)
metrica_fn = len(false_negatives_spacy)
```

```
# Obtenim les mètriques complexes
precisio = metrica_tp / (metrica_tp + metrica_fp)
reclam = metrica_tp / (metrica_tp + metrica_fn)
f1 = 2 * precisio * reclam / (precisio + reclam)

print("DETECTADES PER SPACY EN COMPARACIÓ A L'ETIQUETAT COMPLET")
print("Autèntics positius:", metrica_tp)
print("Falsos positius:", metrica_fp)
print("Falsos negatius:", metrica_fn)
print("==========")
print("Precisió: {}%".format(int(precisio * 10000) / 100))
print("Reclam: {}%".format(int(reclam * 10000) / 100))
print("Mesura F: {}%".format(int(f1 * 10000) / 100))

# Iterem per les etiquetes per obtenir les mètriques individualitzades.
for etiqueta in etiquetes_propies:

    # Obtenim les mètriques pel sistema original de spacy.
    en_resultat_etiqueta = set(filter(lambda e: e[0] == etiqueta,
        entitats_nomenades_res_tuples))
    en_test_etiqueta = set(filter(lambda e: e[0] == etiqueta,
        entitats_nomenades_test_tuples))

    # Operem amb els conjunts per obtenir la intersecció i subtraccions.
    true_positives_etiqueta = en_resultat_etiqueta & en_test_etiqueta
    false_positives_etiqueta = en_resultat_etiqueta - en_test_etiqueta
    false_negatives_etiqueta = en_test_etiqueta - en_resultat_etiqueta

    # Obtenim les mètriques de cada categoria.
    metrica_tp = len(true_positives_etiqueta)
    metrica_fp = len(false_positives_etiqueta)
    metrica_fn = len(false_negatives_etiqueta)

    # Obtenim les mètriques complexes.
    precisio = metrica_tp / (metrica_tp + metrica_fp)
    reclam = metrica_tp / (metrica_tp + metrica_fn)
```

```
    f1 = 2 * precisio * reclam / (precisio + reclam)

    print("= ETIQUETA", etiqueta, "===")
    print("Autèntics positius:", metrica_tp)
    print("Falsos positius:", metrica_fp)
    print("Falsos negatius:", metrica_fn)
    print("==========")
    print("Precisió: {}%".format(int(precisio * 10000) / 100))
    print("Reclam: {}%".format(int(reclam * 10000) / 100))
    print("Mesura F: {}%".format(int(f1 * 10000) / 100))
    print("")
```

Bloc de codi utilitzat per crear les llistes d'entitats nomenades en format *.csv*:

```
import csv, pickle

from utils import EntitatNomenada

# Llegim les variables resultants d'executar les regex
with open('entitats_nomenades.corpus_raw.pickle', 'rb') as file:
    entitats_nomenades_res = pickle.load(file)

# Ordenem les llistes pel seu índex de token
entitats_nomenades_res.sort(key=lambda x: x.inici)

# Exportem els resultats a un arxiu CSV
with open('entitats_nomenades_TAO.csv', 'w', newline='') as csvfile:

    writer = csv.writer(csvfile, quoting=csv.QUOTE_MINIMAL)

    for en in entitats_nomenades_res:

        writer.writerow([" ".join(en.tokens), en.etiqueta])
```