Grau en Enginyeria Informàtica

Especialitat en Computació

Treball de Fi de Grau

Construcció d'un dataset per a l'extracció de models de processos de negoci a partir de textos

Guillem Pla Bertran

Director
Josep Carmona Vargas

Codirector
Lluís Padró Cirera

Tutora del GEP
Olga Pons Peregort

11/06/2022

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Resum

La definició formal de processos de negoci en estàndards com BPMN 2.0 és un pas crucial per a la transformació digital de les organitzacions. Moltes organitzacions tenen els seus processos documentats en text, i crear-ne els models formals té un elevat cost de personal especialitzat.

L'objectiu del projecte és crear un conjunt de dades anotades que relacionin textos amb els models formals que els descriuen, per tal que puguin ser usades per entrenar sistemes neuronals de *Deep Learning* per fer aquesta tasca.

Abstract

Formally defining business processes in standards such as BPMN 2.0 is a crucial step in the digital transformation of organizations. Many organizations have their processes documented in text, and creating formal models has a high cost of specialized staff.

The goal of the project is to create a set of annotated data that relate texts to the formal models that describe them, so that they can be used to train Deep Learning neural systems to do this task.

Índex

Índex de figures

Índex de taules

1

Introducció

El diccionari de l'Institut d'Estudis Catalans [1] defineix un procés com una manera de descabdellar-se una acció progressiva. És a dir, que qualsevol conjunt d'accions que realitzem en el nostre dia a dia es pot definir com a un procés. De la mateixa manera, en una empresa es duen a terme una llarga llista d'accions que permeten aconseguir els objectius marcats.

Evidentment, en una empresa els processos no són tan simples com els que realitzem nosaltres diàriament. En conseqüència, és molt important que cada una de les tasques estigui ben definida.

La definició formal de processos de negoci en estàndards com BPMN 2.0 és un pas crucial per a la transformació digital de les organitzacions, ja que permet gestionar els processos que es duen a terme en una empresa. Moltes organitzacions tenen els seus processos documentats en text, i crear-ne els models formals té un elevat cost de personal especialitzat.

Automatitzar la creació de diagrames BPMN seria molt útil per a les empreses. Una manera per a aconseguir això seria utilitzar models de *Deep Learning*. Malauradament, per entrenar aquests models es necessita una gran quantitat de dades que no sempre és fàcil d'aconseguir. Per evitar l'escassetat de dades, en aquest treball es pretén crear un programa que generi un *dataset* de diagrames BPMN enllaçats amb la corresponent descripció en llenguatge natural.

2

Context

Aquest treball de fi de grau (TFG) del Grau en Enginyeria Informàtica (GEI) de l'especialitat de Computació es fa amb modalitat A (centre), i es realitza en la Facultat d'Informàtica de Barcelona (FIB). El projecte pretén aprofundir en la recerca per a aconseguir una millora en la creació de models BPMN a partir de textos en llenguatge natural.

2.1 Actors implicats

Els actors implicats en aquest projecte són el personal del projecte, és a dir, els directors Josep Carmona Vargas i Lluís Padró Cirera, la tutora del GEP Olga Pons Peregort i l'autor Guillem Pla Bertran.

També hi estan implicades les persones a les quals va dirigit el projecte. Aquestes són totes les persones que fan recerca en el camp de NLP o els que treballen amb BPMN en el món empresarial, principalment *Business Analysts*. Segons la Northeastern University [2] aquestes persones són les responsables d'entendre què es fa en el negoci, quins passos cal seguir per a completar els objectius, avaluar els processos per eficiència, cost i resultat, desenvolupar plans del projecte, entre altres coses.

2.2 Business Process Model and Notation

BPMN vol dir Business Process Model and Notation, en català Model i Notació de Processos Empresarials, és un estàndard per a la modelització de processos empresarials[3]. Aquest proporciona una notació gràfica per descriure un procés en un Business Process Diagram (BPD). Està basat en una tècnica de diagrama de flux molt similar als diagrames d'activitats de l'Unified Modeling Language (UML).

L'objectiu de BPMN és donar suport a la gestió de processos empresarials, tant per als usuaris tècnics com per als usuaris empresarials, per això proporciona una notació

intuïtiva, però que alhora és capaç de representar una semàntica de processos complexa.

Recentment, la quantitat d'empreses que han començat a utilitzar BPMN ha crescut molt. Segons una enquesta del 2018 [4] el 64% dels negocis estan interessats en utilitzar aquest mètode per a simplificar els seus processos empresarials per a poder estalviar diners i millorar la productivitat.

A part de millorar l'eficiència hi ha altres motius que expliquen el perquè és necessari [5]. Augmenten la satisfacció dels clients, milloren la capacitat de resposta organitzativa i ajuden a complir amb noves regulacions, entre d'altres.

2.2.1 Elements de BPMN

Els models BPMN fan servir diagrames simples construïts a partir d'un conjunt d'elements gràfics. Els cinc elements bàsics que es tindran en compte en el projecte són els *Events*, les *Activities*, les *Gateways*, els *Connectors* i els *Swimlane*.

Esdeveniments

Un *Event* o Esdeveniment es representa amb un cercle i significa que alguna cosa passa durant el transcurs d'un procés. Poden tenir símbols que determinin el tipus d'esdeveniment que són. Principalment, es fan servir tres tipus d'esdeveniments:

- **Start event** o Esdeveniment inicial: És el que inicia tot el procés. Tots els processos comencen amb un esdeveniment d'aquest tipus.
- **Intermediate event** o Esdeveniment intermedi: Representa que alguna cosa passa entre l'inici i el final del procés.
- **End event** o Esdeveniment final: Representa el resultat final del procés. Tot procés té com a mínim un esdeveniment final.

Figura 2.1: Representació dels diferents tipus d'Esdeveniments. Font: Elaboració pròpia.

Activitats

Tal com explica Visual Paradigm [6], una *Activity* o Activitat és un "treball"que una empresa duu a terme en un procés de negoci. Les activitats poden ser atòmiques (Tasques) o descomponibles (Subprocessos). N'hi ha tres tipus bàsics:

- **Task** o Tasca: És una activitat atòmica dins d'un flux de processos. Es creen quan l'activitat no es pot desglossar a un nivell més detallat. En general, una persona o aplicacions duran a terme la tasca quan s'executi.
- **Sub-Process** o Subprocés: És una activitat composta que representa una col·lecció d'altres tasques i subprocessos. Per facilitar les comunicacions, no es vol que un diagrama de processos empresarials sigui complex. Mitjançant l'ús de subprocessos, és possible dividir un procés complex en diversos nivells, la qual cosa permet centrar-se en una àrea determinada d'un diagrama de procés.
- **Call Activity** o Activitat de trucada: És una activitat definida en un procés extern a la definició del procés actual. Permet crear una definició de procés que pot ser reutilitzada en altres definicions.

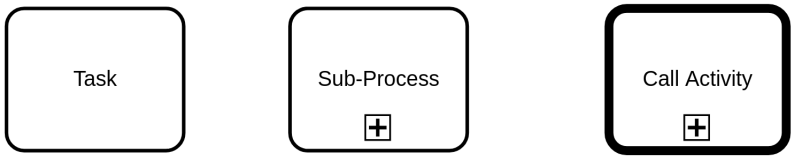

Figura 2.2: Representació dels diferents tipus d'Activitats. Font: Elaboració pròpia.

Gateways

Les *Gateways* o Portes d'entrada determinen quin camí s'empra a través d'un procés que controla el flux tant de fluxos de seqüències divergents com convergents. És a dir, una única porta podria tenir múltiples entrades i múltiples fluxos de sortida en els quals es permet o no l'entrada o la sortida. N'hi ha de diversos tipus, a continuació se n'explica tres dels més importants:

- **Exclusive Gateway** o Porta exclusiva: Se segueix un únic camí.
- **Parallel Gateway** o Porta paral·lela: Se segueixen tots els camins.
- **Inclusive Gateway** o Porta inclusiva: Se segueix un camí o més.

Figura 2.3: Representació dels diferents tipus de *Gateways*. Font: Elaboració pròpia.

Connectors

Els elements anteriors es connecten entre si en un diagrama per crear l'estructura bàsica d'un procés empresarial. Hi ha tres objectes de connexió que proporcionen aquesta funció. Aquests connectors són:

- **Sequence Flow** o Flux de seqüències: S'utilitza per mostrar l'ordre (la seqüència) en què es realitzaran les activitats en un procés.
- **Message Flow** o Flux de missatges: Representa el flux d'informació a través dels límits de l'organització. Els grups, les activitats i els esdeveniments de missatges es poden associar amb el flux de missatges. El flux de missatges es pot personalitzar amb un sobre que mostri el contingut del missatge.
- **Association** o Associació: Les anotacions permeten afegir més informació al diagrama que ajudi a documentar el procés.

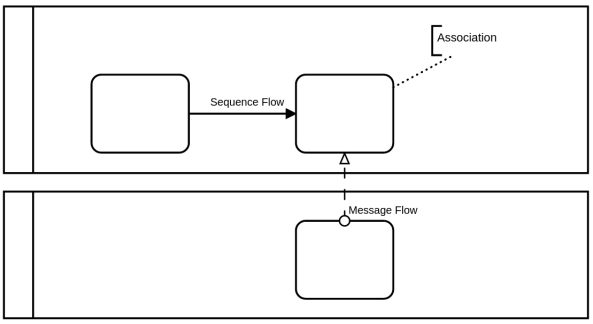

Figura 2.4: Representació dels diferents tipus de Connectors. Font: Elaboració pròpia.

Swimlane

En BPMN existeix un concepte anomenat *Swimlane*, aquest es divideix en dos tipus els *Pool* (en català, piscina), i el *Lane* (en català, carril). Un *Pool* representa un actor que pren part activa en un procés. Es representa amb un contenidor rectangular que pot contenir elements com els descrits anteriorment. Un *Lane* és una subdivisió del *Pool*, es fa servir per categoritzar activitats segons el rol o la funció de l'actor. En la figura 2.5 el *Bank* és el *Pool* i conté dos *Lane* que representen els rols de *Director* i de *Sales Department*.

Figura 2.5: Representació d'una *Swimlane*. Font: Elaboració pròpia.

2.3 Natural Language Processing

En aquest projecte s'utilitzen eines de processament del llenguatge natural o *Natural Language Processing* (NLP). Aquestes eines permeten entendre millor les tasques dels BPMN. A continuació s'explica què és i en què consisteix.

El *Natural Language Processing* és la disciplina informàtica que s'encarrega de tractar computacionalment els llenguatges humans [7]. És un subcamp de la lingüística (ciència que estudia el llenguatge natural [8]), les ciències de la computació i la intel·ligència artificial. Aquest camp s'ocupa, principalment, de processar i analitzar dades del llenguatge natural.

L'abast d'aquesta disciplina és molt ampli. S'aplica en traducció automàtica, generació de llenguatge, anàlisis de sentiments, *chatbots*, anàlisi morfològica, correcció ortogràfica, entre molts d'altres [9]. En aquest treball s'utilitza l'anàlisi morfològica i la generació de llenguatge.

Per a convertir el llenguatge humà en trossos de text llegibles per una màquina, sovint, es fa servir unes anàlisis sintàctiques i semàntiques [10].

L'anàlisi sintàctica o *parsing* és el procés d'analitzar una cadena de caràcters, ja sigui en llenguatge natural, llenguatges informàtics o estructures de dades, d'acord amb les normes d'una gramàtica formal [11]. S'identifica l'estructura del text, les relacions entre les paraules i es representen en un *parse tree* [10].

L'anàlisi semàntica és una tècnica capaç d'analitzar el significat d'un text. Es tracta d'estudiar què volen dir les paraules en conjunt, no només una única paraula [12]. Com que el llenguatge humà és ambigu i polisèmic aquesta tasca no és gens fàcil de dur a terme [13].

A continuació es fa un llistat d'algunes de les subtasques de l'anàlisi sintàctica i l'anàlisi semàntica [10]:

- Convertir en *tokens*: Consisteix a partir una cadena de paraules en unitats útils semànticament anomenades *tokens*.
- Etiquetatge de part de la parla o *Part-of-speech tagging* (PoS): Consisteix a classificar cada *token* una etiqueta. Algunes etiquetes que sovint s'utilitzen en PoS són verb, adjectiu, nom, pronom, conjunció, preposició, etc.
- Anàlisi de l'estructura sintàctica: Es tracta d'aconseguir un arbre que representi tota l'estructura sintàctica d'una frase. Es fa identificant la gramàtica de l'oració.
- Extracció dels lemes: Per a fer que un ordinador sigui capaç d'entendre un text fàcilment cal obtenir l'arrel de les paraules. Aquesta arrel és la que apareix als diccionaris i s'anomena lema.

Hi ha diverses eines i plantejaments que ajuden a desenvolupar projectes de NLP.

El llenguatge de programació *Python* conté una llarga llista de llibreries que poden servir per a fer certes tasques de NLP. Moltes d'elles es poden trobar al *Natural Language Toolkit* (NLTK) [14]. NLTK és una plataforma per a construir programes de Python que treballen amb llenguatge humà i inclou llibreries que executen tasques com les que s'han mencionat anteriorment [15].

Una altra eina és *Freeling*, aquesta és la que ha sigut escollida per al projecte. També fa les tasques esmentades abans i s'ha considerat la més adequada per a fer-la servir. En l'apartat 9.1.2 s'explica les seves principals característiques.

2.4 Natural Language Generation

En aquest projecte s'utilitzen eines de generació de llenguatge natural o, en anglès, *Natural Language Generation* (NLG). Aquestes eines permeten donar un resultat més realista al text final. En aquest apartat s'exposa què és aquesta tecnologia i com es fa servir.

NLG és un subcamp de l'intel·ligència artificial i de la lingüística computacional. S'ocupa de la construcció de sistemes informàtics que poden produir textos comprensibles pels humans a partir d'una font d'informació no lingüística [16].

El seu objectiu principal és convertir la informació que contenen els ordinadors en un escrit entenedor i automatitzar l'escriptura de narratives orientades a les dades [17].

Les aplicacions de NLG utilitzen regles basades en la morfologia, el lèxic, la sintaxi i la semàntica per a escollir com redactar les respostes de forma adequada. Aquest procés es fa en tres etapes [18]:

- Planificació del text: Es formula el contingut general i s'ordena de forma lògica.
- Planificació de les frases: S'organitzen les frases tenint en compte els signes de puntuació i afegint pronoms o conjuncions quan sigui necessari.
- Realització: Es genera la frase considerant la precisió gramatical i assegurant que es respecten les normes de puntuació i conjugació.

Existeixen tres tipus d'eines de *Natural Language Generation* [17]:

- Bàsic: Només transforma dades en text.
- Basat en plantilles: Encaixa les dades en plantilles de text.
- Avançat: Dedueix característiques a partir de les dades i utilitza el context per a adaptar el missatge.

Es poden trobar similituds entre *Natural Language Generation* (NLG), *Natural Language Processing* (NLP) i *Natural Language Understanding* (NLU), ja que totes tres treballen la mateixa disciplina. Tanmateix, existeixen matisos que les diferencien.

El programari NLG és capaç d'escriure, però no pot llegir. Això el diferencia de NLP que llegeix dades desestructurades de llenguatge humà i les converteix en dades estructurades que es poden entendre. Aquesta part de NLP s'anomena *Natural Language Understanding* [19].

NLG i NLU són subseccions d'un domini de NLP més general que compren tot el programari que interpreta o produeix llenguatge humà. NLP s'encarrega d'entendre les dades en funció de la gramàtica i el context. NLP converteix text en dades estructurades i

NLG genera un escrit basat en dades estructurades [19].

Aquesta tecnologia és tan versàtil que s'utilitza en molts camps. Els principals camps són generació automàtica d'informes, subtítols d'imatges, *chatbot*, escriptura creativa i humor computacional [16]. Però, també es fa servir en disciplines com política, banca i finances, fabricació i assegurances [20].

Hi ha diverses eines que es poden fer servir per a desenvolupar projectes de NLG. Es poden diferenciar en eines comercials i eines de codi lliure. A continuació, es mostren exemples de cada tipus amb una breu descripció [19].

Algunes eines comercials són:

- *Arria NLG*: És una de les empreses líders mundialment en tecnologies NLG. El seu programari és una forma d'intel·ligència artificial que transforma dades estructurades en llenguatge natural [21].
- *AX Semantics*: Aquesta companyia ofereix serveis de NLG en camps com el comerç electrònic, periodisme i dades. Utilitza IA i *machine learning* per a entrenar el motor NLP.
- *Wordsmith*: És un motor NLG que permet als usuaris convertir dades en text en qualsevol format o escala.

Algunes llibreries de codi obert són:

- *NaturalOWL*: És un conjunt d'eines de codi obert que s'utilitza per a generar descripcions sense fer gaire programació.
- *GPT-3*: És un model de llenguatge autoregressiu que fa servir *Deep Learning* per produir textos que simulen la redacció humana [22]. Ha estat creat per l'empresa *OpenAI*.
- *SimpleNLG*: És un dels realitzadors de codi oberts més utilitzats. Aquesta API de Java ha sigut desenvolupada pel fundador de *Arria*. Les seves funcionalitats són bàsiques, però són senzilles de fer anar i estan ben documentades. És l'eina que s'ha decidit fer servir en aquest treball, en l'apartat [23] se'n parla més a fons.

2.5 Problema a resoldre

Com s'ha comentat anteriorment, el modelatge dels processos empresarials és una part vital a l'hora d'establir el funcionament d'un projecte. Moltes empreses tenen textos que expliquen com estan definits les accions a realitzar, però no ho tenen modelat en un llenguatge formal. El modelatge pot arribar a comportar fins al 60% del temps gastat en els projectes de gestió de processos.

Una possible solució per a automatitzar la creació dels diagrames a partir de descripcions en llenguatge natural és fer servir tècniques modernes d'Intel·ligència Artificial. En aquest cas concret es podria entrenar un model de *Deep Learning* a partir d'un conjunt de diagrames que tenen associat la seva descripció en llenguatge natural.

Lamentablement, tal com Matthew Stewart [24] explica, un dels reptes de l'aprenentatge automàtic és la manca de dades i la falta de bones dades. Aconseguir els conjunts de dades adequats és complex, ja que per motius de privacitat i confidencialitat molt poques empreses alliberen les descripcions dels seus processos.

En aquest projecte es pretén abordar el problema de l'escassetat de dades. Es farà generant descripcions sintètiques, que semblin mínimament reals, i que permetin entrenar algorismes de forma fiable i controlada.

A continuació, en la figura 2.6 es mostra un exemple d'un model BPMN que representa el procés de preparació d'un àpat. Per a il·lustrar el que es pretén fer al treball el model està acompanyat d'una descripció en llenguatge natural generat a mà, però l'objectiu és aconseguir generar una descripció similar automàticament.

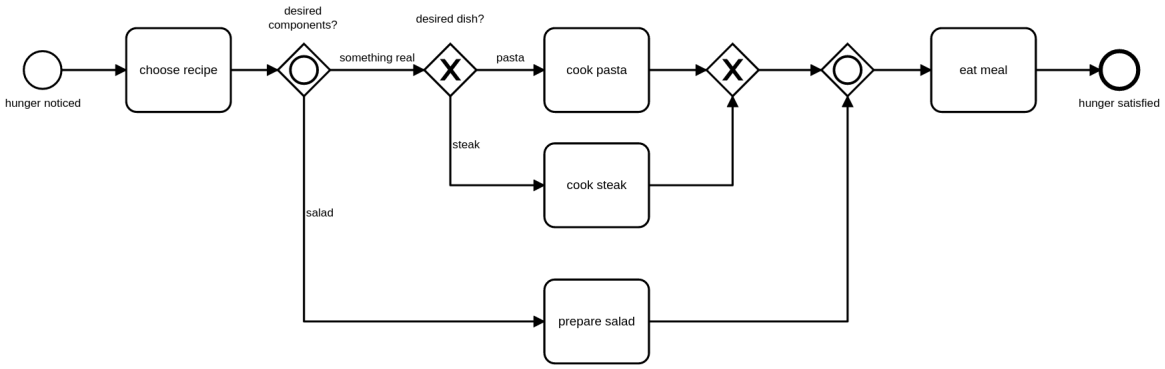

Figura 2.6: BPMN d'exemple. Font: camunda.com

En aquest cas la descripció de la figura 2.7 és senzilla perquè l'exemple utilitzat ho era, però no sempre serà així. També s'hauria de vigilar que no sembli un text molt artificial, però això ja és més complicat. En la figura 2.8 es pot observar una descripció més propera al llenguatge utilitzat per un humà. Aquest hauria de ser l'objectiu final a assolir de cara a tenir un bon conjunt de dades, però no és tan senzill d'aconseguir.

When hunger is noticed, a recipe is chosen. Then the question 'desired components?' is asked.
If the answer is salad, then a salad is prepared.
If the answer is something real, then the question 'desired dish?' is asked.
If the answer is pasta, then pasta is cooked, otherwise if the answer is steak, then a steak is cooked.
Then the meal is eaten and hunger is satisfied.

Figura 2.7: Descripció robotitzada del BPMN d'exemple de la figura 2.6. Font: Elaboració pròpia.

Once the hunger is noticed, a recipe is chosen.
If salad is chosen as a desired component, then a salad is prepared.
Besides of that, it is possible to select something real.
In that case, the desired dish mush be chosen between pasta and steak.
If the desired dish is pasta, then pasta is cooked, otherwise a steak is cooked.
Finally, the meal is eaten and hunger is satisfied.

Figura 2.8: Descripció més natural del BPMN d'exemple de la figura 2.6. Font: Elaboració pròpia.

En l'exemple 2.8 podem veure que s'hi ha afegit una sèrie de connectors per a fer-lo menys robotitzat. Cal tenir en compte que l'objectiu del projecte és generar un conjunt de dades que es pugui fer servir per a entrenar un model d'aprenentatge automàtic. Això implica que si tots els registres contenen construccions similars, el model aprendrà poc. Per això, s'hauria d'intentar fer servir sinònims o connectors equivalents per a cada registre.

A més, un model pot comptar amb moltes més activitats i ramificacions. Això pot complicar la generació de la frase per què cal decidir quan s'ha de fer un nou paràgraf. També cal estructurar bé el text perquè quan diverses ramificacions s'ajuntin s'entengui el conjunt sencer del paràgraf. Aquests reptes són els que s'intentaran superar en el transcurs d'aquest projecte.

En la figura 2.9 s'hi mostra un exemple d'un BPMN real i més complex. En la figura 2.10 s'hi pot veure una possible descripció ideal feta a mà. Malgrat estar escrita per un humà, es pot veure com en alguns casos costa de seguir, sobretot quan hi ha bifurcacions, frases molt llargues o activitats que assenyalen a altres activitats esmentades anteriorment. Si fins i tot costa que una persona faci una descripció acurada i fàcil d'entendre d'un model complex, un programa automàtic també ho tindrà complicat.

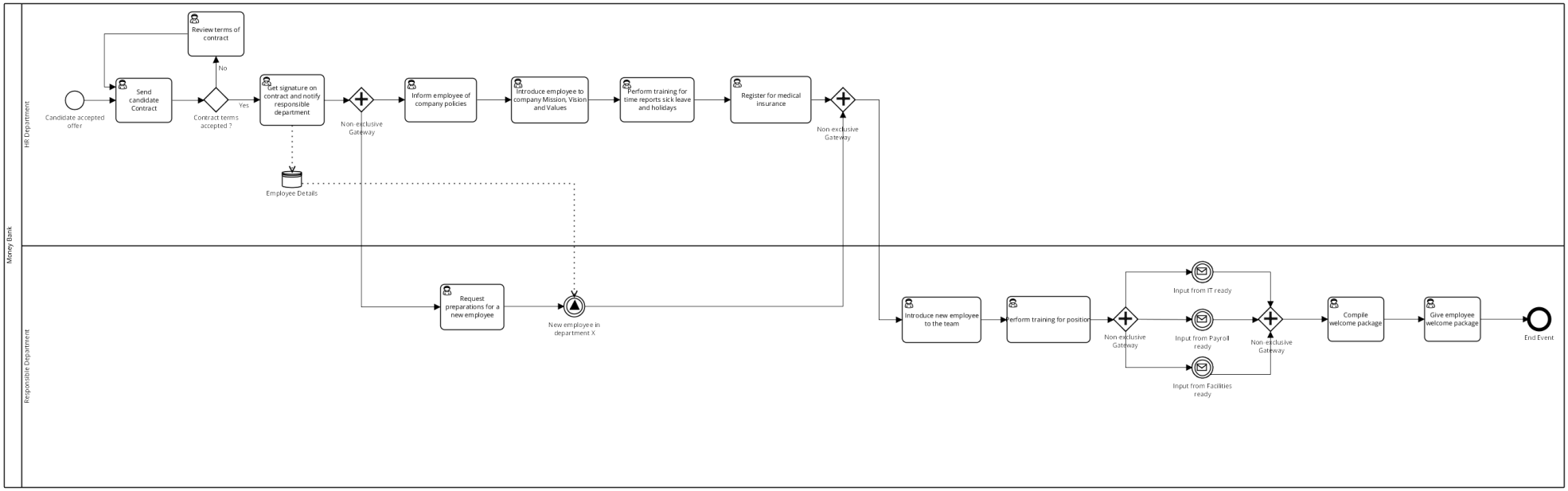

Figura 2.9: Exemple d'un BPMN més complex. Font: bpmn-miwg

The process starts once the candidate has accepted the offer, then the HR Department sends him the contract.
If the contract terms aren't accepted, then the HR Department reviews the terms of the contract and sends it again.
If the contract terms are accepted, then get signature on the contract and notify the responsible department.
Then the HR Department informs the employee of company policies, introduces him to company mission, vision and values, performs training for time reports, sick leave and holidays and registers him for medical insurance.
At the same time, the responsible department requests the preparations for a new employee, the event new employee in department X happens.
Once the previous activities are done, the responsible department introduces the new employee to the team and performs training for the position.
Then, at the same time, the events input from IT ready, input from Payroll ready, input from facilities ready happen.
Once the previous activities are done, the responsible department compiles the welcome package and finally gives him he welcome package.

Figura 2.10: Descripció del BPMN de la figura 2.9. Font: Elaboració pròpia.

3

Justificació

Donat que el modelatge de processos amb l'estàndard BPMN és molt específic l'estat de l'art no és molt extens. Tot i això, es poden trobar diverses aplicacions similars, encara que no fan ben bé el que es pretén en aquest treball. Aquests projectes fets anteriorment els podem classificar entre els que transformen d'un diagrama BPMN a text natural i els que ho fan a l'inrevés.

Abans, desenvolupar una eina que automatitzés qualsevol procés requeria molt esforç i dedicació, ja que, s'havien de tenir en compte totes les possibles combinacions en els possibles *inputs*. Això feia que en certs camps, com ara el de convertir un text natural, fos pràcticament impossible d'aconseguir. Per sort, mètodes com el *Deep Learning* permeten generar contingut de forma ràpida i eficient. Però, per aconseguir un model funcional es requereix un conjunt de dades molt gran i divers, i això no sempre és fàcil d'aconseguir.

Per a solucionar el problema plantejat en aquest treball no disposem d'un bon *dataset* que ens permeti generar un model de *Deep Learning* fiable. És per això, que la part principal d'aquest treball consisteix a crear una aplicació que generi un conjunt de dades variat i amb prou registres.

Això es farà convertint diversos diagrames a un llenguatge natural que sigui entenedor per qualsevol persona sense coneixement de BPMN. En aquest cas s'ha decidit que es farà servir l'anglès, ja que és un dels més utilitzats al món.

Per a fer aquesta primera part, hi ha un treball del 2016 titulat *Transformant Models BPMN a Llenguatge Natural* [25] que fa quelcom semblant. En el treball es llegeix el fitxer BPMN, després es crea un RSPT (Refined Program Structure Tree) per a guardar la informació necessària per solucionar el problema i també es fa servir una xarxa de Petri per a extreure l'estructura del procés. Finalment, es fa la generació del text analitzant cada element i concatenant els diferents missatges seguint l'estructura dels arbres creats anteriorment.

La segona part del projecte consisteix a entrenar una xarxa neuronal que sigui capaç d'interpretar una descripció en llenguatge natural i que ho converteixi en un model BPMN. Per a fer això, es farà servir el dataset que hem creat a la primera part. De treballs anteriors i aplicacions existents que converteixin de text a diagrames n'hi ha algunes que s'esmenten a continuació. Però cap d'elles fa servir un model de *Deep Learning*. Per tant, en aquest cas caldrà entrenar el model des de zero.

Process Talks [26] és una aplicació web que permet modelar processos a partir d'una explicació informal, sigui escrivint o parlant. L'eina fa servir regles i patrons que analitzen l'estructura sintàctica de les frases i treuen informació de qui realitza cada acció, o quina acció precedeix una altra. Aquesta implementació és costosa de mantenir i no és escalable, ja que, és complicat traspassar-la a nous idiomes. Per això, l'objectiu del projecte és crear una alternativa basada en *Machine Learning*.

A part d'aquesta aplicació web, també podem trobar un treball de recerca anomenat *Process Model Generation from Natural Language Text* [27]. Podem veure que es combinen algunes eines existents en el processament del llenguatge natural per a obtenir uns millors resultats. L'avaluació mostra que per a un conjunt de 47 parells de text-model de la indústria i els llibres de text, generen de mitjana el 77% dels models correctament.

Com s'ha pogut veure hi ha poques implementacions que serveixin de cara a plantejar una solució. Tanmateix, s'intentarà aprofitar el màxim de recursos disponibles per a reduir costos i temps.

4

Abast

4.1 Objectius

El principal objectiu del projecte és crear un conjunt de dades anotades que relacionin textos amb els models formals que els descriuen. Les dades han de servir per a ser usades per entrenar sistemes neuronals que donats un text en llenguatge natural el converteixin a un diagrama BPMN.

Com ja s'ha dit anteriorment a l'apartat de Justificació, el projecte es pot dividir en dos parts. La primera i principal és la **Generació del Dataset** i la segona és la d'**Entrenar un model** de Deep Learning com a demostració que el *dataset* és útil. En la figura 4.1 es pot veure les diferents parts del projecte i els subobjectius que té cadascuna d'elles.

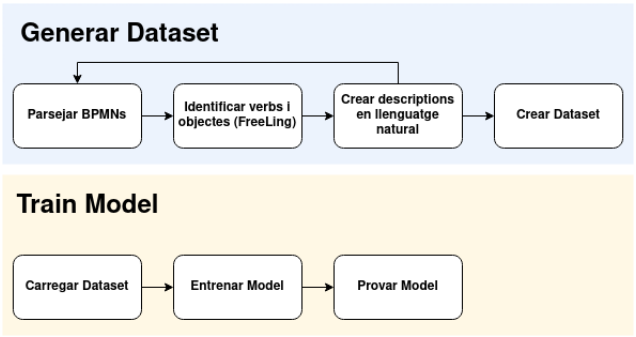

Figura 4.1: Subobjectius de cada part del projecte. Font: Elaboració pròpia.

En fase de generació del dataset el que es busca és tenir un conjunt de dades amb suficients registres i amb descripcions prou variades que serveixi per a poder entrenar després un model.

Per a generar les dades cal crear un programa que llegeixi (en anglès *parse*) els elements d'un diagrama BPMN i emmagatzemi la informació necessària en un fitxer JSON. Caldrà

extreure i modificar els elements que es necessitin. Un cop fet això només caldrà fer la tria dels atributs que es volen. Després s'identificaran els verbs, objectes i complements de les frases de cada element utilitzant un software d'anàlisis lingüístic anomenat *FreeLing* [28] [29] [30] [31] [32]. Tot seguit, es procedirà a crear les descripcions en llenguatge natural, utilitzant un software de generació de llenguatge natural anomenat *SimpleNLG*[23].

Un cop s'hagi desenvolupat el programa anterior, simplement caldrà repetir el procés de generació de text per a cada un dels models BPMN i emmagatzemar el conjunt de les dades en un format adequat per a poder-ho utilitzar després.

La segona part del projecte és la de crear un model de *Deep Learning* que serveixi com a demostració que el conjunt de dades generat és utilitzable. En aquest pas s'haurà de carregar el dataset generat anteriorment, entrenar el model i provar que retorna el resultat que es vol. Si no s'ha obtingut un resultat satisfactori, es pot polir el model canviant paràmetres.

4.2 Possibles obstacles

És molt comú que quan es comença a fer un projecte es trobin obstacles i problemes que no s'havien plantejat. Quan això passa pot ser que el projecte s'endarrereixi molt. Per tant, s'ha d'intentar tenir en compte el conjunt de riscos que poden sorgir.

Un dels obstacles que pot aparèixer és que no es disposi de prou diagrames BPMN i, per tant, no poder crear un *dataset* amb gaires registres. També pot ocórrer que la utilització de llibreries no sempre funcioni correctament. En aquest cas s'haurà d'invertir molt més temps a crear alternatives. Tenir suficient informació disponible per a fer un projecte és important, i el fet que no es pugui trobar prou documentació sobre un tema és un risc a tenir en compte. A més a més, s'ha de tenir present que el temps és limitat, per tant, és possible que no sempre es puguin assolir tots els objectius marcats.

5

Metodologia

5.1 Mètode Kanban

Per a dur a terme aquest treball s'ha decidit fer servir la metodologia *Kanban*. Tal com s'explica en la web APD [33], és un sistema de producció efectiu i eficient que està dins del grup de metodologies àgils [34]. La base de les metodologies *Agile* és permetre una planificació iterativa, això implica descobrir requisits i desenvolupar solucions mentre s'avança en el desenvolupament. El principal tret de *Kanban* és l'ús de targetes visuals que ens ajuden a organitzar les tasques.

Les targetes estan definides en una web de gestió de tasques. I aquests es poden moure per definir el seu estat. S'han fet servir els següents estats:

- **To do**: Significa que la tasca encara s'ha de realitzar.
- **Doing**: Implica que la tasca s'està duent a terme.
- **Review**: Vol dir que la tasca s'ha realitzat, però el resultat pot no ser definitiu.
- **Done**: Significa que la tasca ja es dona per acabada

A més a més, la metodologia es basa a acabar la feina que ja s'ha començat i en evitar tenir diverses tasques a mitges. També permet prioritzar cada tasca, per a intentar fer de la millor forma aquelles funcionalitats que es consideren més importants.

5.2 Eines

En aquest apartat s'explica el conjunt d'eines que permetran aplicar la metodologia *Kanban* al projecte.

Figura 5.1: Metodologia Kanban. Font: Pixabay.

Per començar es necessita un controlador de versions [35], això és un software que permet guardar i gestionar una còpia de les diferents versions del programa. S'ha escollit *Git* [36] en combinació amb *GitHub* [37], què és un servei que web què permet fer ús del control de versions en línia.

Per a comprovar que el desenvolupament de les tasques es fa correctament es fa servir un servei web d'organització i gestió de tasques anomenat *Todoist* [38]. Aquest software permet definir les diferents activitats que cal acomplir en cada part del projecte i posarles en el taulell *Kanban*. També es pot definir una data límit per a cadascuna d'elles. Així doncs, es pot saber si portem el desenvolupament al dia.

5.3 Validació

El software es revisa cada una o dos setmanes amb els directors del treball, de manera que en cas d'haver-hi algun error o dubte es pot decidir com es procedeix de forma ràpida. A més, aquesta metodologia també permet proposar canvis en el disseny del programa, de manera que es poden afegir o eliminar funcionalitats en funció del temps restant per a l'entrega.

6

Planificació temporal

Un dels aspectes més importants a l'hora de dur a terme un projecte és el temps del qual es disposa. En el moment en què es redacta aquest document la data per a la finalització del treball encara no està definida, de totes maneres, es fixa el límit per acabar el projecte a principis del mes de gener del 2022. El treball es va començar el 10 de juliol del 2021. Això vol dir que es disposen de 184 dies en total per a fer el treball, inclosos caps de setmana i festius. S'estima que en un Treball de Fi de Grau (TFG) la dedicació total és de 540 h. Això implica que s'haurien de destinar unes 3 h al dia per a complir amb els objectius establerts. Evidentment, no cada dia es treballarà aquestes hores, però sí que s'haurien de fer unes 21 h setmanals.

6.1 Definició de tasques

En aquest projecte s'ha decidit dividir la feina a fer en un conjunt de tasques. Aquesta divisió permet tenir un control més gran del projecte. De manera que es pugui saber l'inici, el final i les dependències que té cada tasca. Això permet saber en tot moment si el projecte està endarrerit o no. I en cas que ho estigui, prendre les mesures oportunes.

A continuació s'expliquen els recursos necessaris per al projecte i totes les tasques i subtasques que s'han definit per al treball.

6.1.1 Recursos necessaris

S'ha determinat que els recursos necessaris per a desenvolupar el projecte són:

Humans - R1

Els recursos humans necessaris són tots dos codirectors del projecte Josep Carmona i Lluís Padró, la tutora del GEP Olga Pons i l'autor del treball o programador Guillem Pla.

Maquinari - R2

El maquinari que es fa servir és un ordinador portàtil de gamma mitjana. Aquest permet tant documentar, com programar, com entrenar el model. A més, permet la llibertat de moviments i, per tant, facilitar les reunions presencials. El portàtil utilitzat és un *Acer Aspire 3*.

Programari - R3

El software que es preveu utilitzar el *GitHub*(controlador de versions), *Todoist*(gestió del projecte), *PyCharm*(programació de projectes en Python), *IntelliJ IDEA*(programació de projectes en Java), *Google Collab*(executar codi al núvol) i *Tensorflow*(llibreria Python de machine learning).

6.1.2 Gestió del Projecte - GP

La primera part del projecte és la de gestionar de forma correcta el projecte. En aquest apartat s'hi engloben les gestions no tècniques que cal fer per a engegar el projecte. El temps total per a realitzar aquesta part s'estima en 90 hores. Aquestes són la contextualització i abast, la planificació temporal, el desenvolupament d'un pressupost, la realització de l'informe de sostenibilitat, l'elecció de la metodologia i la instal·lació del programari necessari. A continuació s'expliquen en més detall.

Contextualització i abast - GP-C

En aquesta tasca es defineixen els fonaments del projecte. Dins d'aquesta s'hi inclou la tria del tema, l'objectiu principal a resoldre, l'explicació de per què és necessari fer aquest treball i l'abast d'aquest. Per tant, una part de la feina a realitzar consisteix a fer reunions amb els directors del treball i una altra en documentar les decisions preses. S'estima que es realitza en unes 15 hores.

Planificació temporal - GP-T

Aquí es tracta de dividir el treball en tasques i d'establir un calendari per a cadascuna d'elles. Això es fa per a poder tenir, en tot moment, el control de la feina feta i la que falta per fer. S'estima que són necessàries unes 13 hores.

Metodologia - GP-M

En aquesta tasca el que es fa és estudiar el conjunt de metodologies de treball disponibles i fer una tria adequada d'una d'elles. Això permetrà seguir el calendari d'una forma més acurada. S'estima que la tasca de Metodologia es pot dur a terme en 13 hores.

Pressupost - GP-P

En aquest apartat es crea un informe on s'expliquen els termes econòmics del treball. D'acord amb les hores realitzades i amb els recursos emprats es fa un càlcul aproximat sobre el cost que té crear el projecte. Aquesta tasca es pot realitzar en unes 13 hores.

Informe de sostenibilitat - GP-S

Aquesta tasca és molt semblant a l'anterior, però en aquest cas, el que s'estudia és l'impacte mediambiental i social que el treball pot tenir. Aquesta tasca s'hauria de desenvolupar en 13 hores.

Instal·lació de programari - GP-I

Abans de començar a programar és vital decidir quines eines de software es faran servir. Per això, en aquesta tasca el que es fa és un estudi d'un conjunt d'eines disponibles al mercat i posteriorment es decideix quines es volen fer servir. Per a decidir entre un programa o un altre es té en compte la facilitat d'ús, si ja s'ha fet servir prèviament, la corba d'aprenentatge, la gratuïtat del programa i si és de codi obert. Un cop ja s'ha fet la tria només queda instal·lar totes les eines escollides. S'estima que es pot tardar unes 20 hores.

6.1.3 Generació del dataset - GD

Un cop s'ha definit el projecte a realitzar (GP-C 6.1.2) ja es pot començar el desenvolupament. La primera part consisteix a generar un conjunt de dades que permeti entrenar un model de Deep Learning. Aquest conjunt de dades constarà d'un conjunt de parelles formades per un model BPMN i un text descriptiu del model. En aquest apartat s'espera invertir unes 190 hores en total. Per a aconseguir això, es necessiten dur a terme les tasques següents.

Obtenir models BPMN - GD-O

Primer de tot, cal obtenir un conjunt de models BPMN que serveixi de base per a poder desenvolupar i testejar el programa. De diagrames BPMN se'n poden trobar gratuïtament a internet, per tant, cal fer una cerca a la xarxa i descarregar-los. També és convenient comprovar que els arxius estan modelats correctament i es poden fer servir. S'estima que la durada d'aquesta tasca és d'unes 13 hores. Aquesta tasca no es pot iniciar fins que no s'hagi contextualitzat el projecte (GP-C).

Parsejar BPMN - GP-P

Després de tenir una sèrie de models descarregats, es procedeix a llegir l'arxiu que els conté, és a dir, a parsejar-los. Per a fer-ho caldrà recórrer el fitxer *.bpmn* i emmagatzemar cada element útil en un fitxer *.json*. S'estima que la durada d'aquesta tasca és d'unes

27 hores. Aquesta tasca no es pot iniciar fins que no s'hagi finalitzat la tasca d'obtenir models BPMN (GP-O).

Identificar verbs - GP-I

La tasca anterior ha permès obtenir el conjunt d'elements que conformen el model. Aquests elements sovint tenen una curta frase explicant en què consisteix la tasca del model. Aquesta frase s'utilitzarà més endavant per a generar un text natural, però abans es necessita l'anàlisi sintàctica d'aquesta frase. Per a fer-ho, es fa servir la llibreria *Freeling* que retornarà el tipus de cada paraula de la frase. Aquesta informació s'emmagatzema per a fer-la servir més endavant. S'estima que la durada d'aquesta tasca és d'unes 47 hores. Aquesta tasca no es pot iniciar fins que no s'hagi finalitzat la tasca de parsejar BPMN (GP-P).

Crear descripcions - GP-D

En aquesta tasca es crea una descripció en llenguatge natural a partir d'un diagrama BPMN. Per a fer-ho cal utilitzar la informació emmagatzemada anteriorment i fer-la servir d'input. Per a crear aquest programa es fa servir una llibreria de generació de llenguatge natural anomenada SimpleNLG. Aquest programa retornarà la descripció del model introduït. S'estima que la durada d'aquesta tasca és d'unes 70 hores. Aquesta tasca no es pot iniciar fins que no s'hagi finalitzat la tasca d'identificació de verbs (GP-I).

Crear dataset - GP-C

Al final, s'utilitzarà el programa creat en la tasca anterior per a processar múltiples diagrames. D'aquesta forma s'obtindrà per a cada model una descripció que servirà per a crear el dataset. Quan s'estigui satisfet amb el nombre de descripcions creades, s'ajunta tot en un mateix arxiu. S'estima que la durada d'aquesta tasca és d'unes 33 hores. Aquesta tasca no es pot iniciar fins que no s'hagi finalitzat la tasca de creació de descripcions (GP-D).

6.1.4 Crear model Deep Learning - MD

La segona part del desenvolupament consisteix a crear un model de Deep Learning que utilitzi el dataset creat anteriorment per a aprendre a descriure un model BPMN automàticament. En aquesta fase és les tasques d'entrenament i provar el model se superposen. Això passa perquè un cop es tingui un model entrenat s'ha de comprovar si funciona bé. En Machine Learning ja se sap que no s'obté un model funcional de seguida, per tant, s'ha de validar el model i en cas que no s'obtingui el resultat desitjat s'haurà de generar un nou model. I així successivament. Per a realitzar el conjunt de tasques d'aquesta secció s'espera destinar unes 112 hores.

Carregar dataset - MD-C

Per a crear un model de Deep Learning primer es carrega el dataset desitjat. Així doncs, s'haurà de llegir el fitxer que conté les dades amb una llibreria adequada. En principi sembla una tasca fàcil, però la lectura de fitxers sempre pot donar problemes. I més quan el conjunt de dades ha sigut autogenerat. S'estima que la durada d'aquesta tasca és d'unes 26 hores. Aquesta tasca no es pot iniciar fins que no s'hagi finalitzat tot l'apartat de Gestió de Projectes (GP) i l'apartat de Generació del Dataset (GD).

Entrenar model - MD-E

A continuació, ja es pot entrenar el model usant les dades carregades. D'entrada, en aquesta tasca s'ha de definir quina arquitectura es vol fer servir en el model. Després es definiran uns paràmetres i es procedirà a entrenar el model predictiu. Si no s'ha obtingut uns bons resultats, es pot canviar els paràmetres amb la intenció de millorar la generació final del text, però no és necessari. S'estima que la durada d'aquesta tasca és d'unes 48 hores. Aquesta tasca no es pot iniciar fins que no s'hagi finalitzat la tasca de carregar el dataset (MD-C).

Provar model - MD-P

Aquesta tasca consisteix a comprovar si els resultats són prou bons. Com que no es tracta d'un procés de classificació automàtic validar que la sortida obtinguda és un procés manual i, per tant, costós. Aquesta tasca es durà a terme pràcticament alhora que la d'entrenar el model. S'estima que la durada d'aquesta tasca és d'unes 38 hores. Aquesta tasca no es pot iniciar fins que no s'hagi finalitzat la tasca d'entrenar un model (MD-E).

6.1.5 Documentació - DO

La documentació és una tasca molt important del treball, ja que és la que permet explicar com s'ha organitzat, realitzat i quins resultats s'ha obtingut. S'ha de dur a terme paral·lelament durant tot el desenvolupament. En aquest apartat s'espera invertir unes 120 hores en total.

Memòria - DO-M

Aquesta tasca es farà alhora que la resta del projecte i contextualitzarà el treball, es detallaran tots els passos que s'han seguit i es justificarà cada decisió presa. S'estima que la durada d'aquesta tasca és d'unes 90 hores.

Presentació - DO-P

En aquesta tasca es crearà una presentació que contindrà un breu resum del projecte elaborat. Aquesta serà un ajut visual de cara a la presentació que es faci d'aquest treball davant del tribunal que avaluï el projecte. També s'hi ha de tenir en compte el temps

per a preparar l'explicació oral. S'estima que la durada d'aquesta tasca és d'unes 30 hores. La tasca no es pot iniciar fins que no s'hagi finalitzat la resta del treball (GP, GD, MD, DO-M).

Codi	**Nom Tasca**	**Temps Estimat**	**Dependències Temporals**	**Recursos**
GP-C	Contextualització i abast	15 h	-	R1, R2
GP-T	Planificació temporal	13 h	GP-C	R1, R2
GP-M	Metodologia	13 h	GP-C	R1, R2
GP-P	Pressupost	13 h	GP-C	R1, R2
GP-S	Informe de sostenibilitat	13 h	GP-C	R1, R2
GP-I	Instal·lació programari	20 h	-	R1, R2
GD-O	Obtenir models BPMN	13 h	-	R1, R2, R3
GD-P	Parsejar BPMN	27 h	GD-O	R1, R2, R3
GD-I	Identificar verbs	47 h	GD-P	R1, R2, R3
GD-D	Crear descripcions	70 h	GD-I	R1, R2, R3
GD-C	Crear dataset	33 h	GD-C	R1, R2, R3
MD-C	Carregar dataset	26 h	GD	R1, R2, R3
MD-E	Entrenar model	48 h	MD-C	R1, R2, R3
MD-P	Provar model	38 h	MD-E	R1, R2, R3
DO-M	Memòria	90 h	-	R1, R2
DO-P	Presentació	30 h	DO-M	R1, R2

Taula 6.1: Taula de les tasques. Font: Elaboració pròpia.

6.2 Estimacions i Gantt

En aquest apartat s'utilitza un diagrama de Gantt per a visualitzar l'ordre i duració de cadascuna de les tasques descrites anteriorment. Un diagrama de Gantt [39] és un tipus de gràfic que il·lustra el calendari d'un projecte. Aquest diagrama mostra les tasques que s'han d'executar en l'eix vertical i en l'eix horitzontal mostra el moment d'inici, el moment de finalització i la duració de cada tasca. Com s'ha comentat anteriorment en l'apartat de Planificació temporal 6 en cada setmana es disposa de 21 h per a executar les diferents tasques. A continuació, es mostra el diagrama de Gantt.

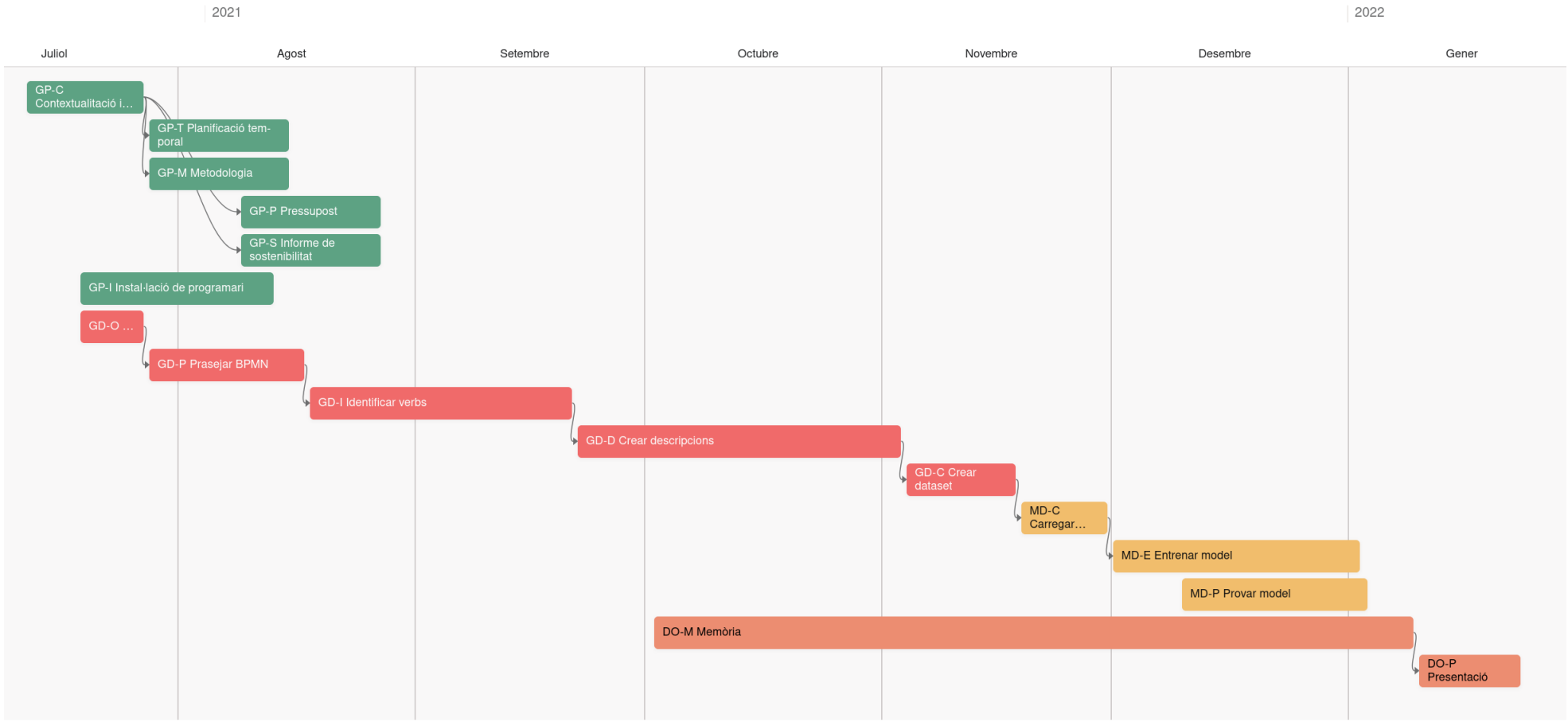

Figura 6.1: Diagrama de Gantt del projecte. Visualitzat en la web Asana. Font: Elaboració pròpia

6.3 Plans alternatius i obstacles

En tot projecte s'ha de tenir previst que puguin aparèixer inconvenients. En conseqüència cal pensar bé quins problemes poden sorgir en cada fase del treball i proposar una solució alternativa. A continuació s'explica quins plans es poden seguir en cas que apareguin alguns obstacles.

- Un dels principals problemes que sorgeix a quasi tots els projectes és la manca de temps. Per abordar aquest obstacle s'ha decidit que no cal crear cap tasca alternativa, sinó que s'intentarà treballar més hores de les proposades per poder arribar a temps a la data límit.

  També s'ha tingut present això a l'hora d'assignar la càrrega a cada tasca. Es pot veure que el temps estimat total per a la realització del projecte és d'unes 509 hores, però en TFG s'haurien de dedicar unes 540 hores. Llavors es té unes 31 hores que es poden realitzar de més i continuar acabant a temps el pla.

  A més, en l'anàlisi de temps dedicat a cada tasca també s'ha tingut en compte que entrenar i provar el model no sortirà bé de seguida. Per tant, en els temps estimats ja té en compte que es pugui tardar a obtenir un resultat desitjable.

- Un altre dels problemes que pot sorgir és que la informació sobre el tema del treball no sigui gaire extensa, això suposa un problema per què costa més entendre com s'ha de procedir. La solució és esprémer al màxim la informació disponible i aprendre a partir de la investigació pròpia.

- Un altre problema que pot sorgir és que el dataset creat no sigui prou divers o que no es tinguin gaires models BPMN per a crear el dataset. Si un d'aquests casos apareix, implicaria que mai es podria obtenir un bon model de Deep Learning. Com que això no és tolerable s'hauria de crear una nova tasca anomenada **Crear models BPMN inventats (GD-O)**. Es tractaria d'usar *data augmentation* per a obtenir més models BPMN i així intentar aconseguir un dataset més divers. Evidentment, això és complicat de realitzar i no garanteix uns bons resultats, però s'hauria de provar. La realització d'aquesta tasca podria durar unes 15-25 hores.

- L'últim possible obstacle que s'ha analitzat és que no es disposi de recursos suficients per a entrenar el model. En aquest cas, la solució proposada és executar el codi en servidors al núvol amb més velocitat de processament que les nostres màquines en local. Aquests podran ser tant de pagament com gratuïts. S'intentarà utilitzar sempre l'opció més econòmica. Si cal fer servir aquests serveis, no comportarà un temps afegit, ja que, la posada a punt d'aquests servidors és molt ràpida i intuïtiva.

6.4 Canvis en la planificació

Durant el transcurs del treball es va decidir endarrerir la data de finalització del treball. Aquesta va passar de ser del gener del 2022 al juny del 2022. Aquest canvi es va produir perquè compaginar el projecte, amb la feina i una assignatura de la universitat va ser impossible. Això implica que s'ha tingut més dies dels que es van calcular inicialment, però el nombre estimat d'hores dedicades totals és el mateix.

També hi va haver un endarreriment en la primera etapa del projecte, la de *parsejar* els models BPMN. En la planificació temporal inicial no es va tenir en compte que aquesta etapa podia tardar tant, ja que semblava una part fàcil del projecte. Es va intentar buscar eines o programes que llegissin els models. Malauradament, l'eina escollida va resultar ser poc fiable i es va haver de començar de nou. A més a més, es va intentar programar de zero un *parser* que llegeixi cada model, però finalment es va decidir utilitzar la llibreria *Camunda*. Els motius d'aquesta decisió s'expliquen més endavant en l'apartat d'Anàlisi d'alternatives 9.2.

Un altre obstacle que es va trobar durant la realització del treball és que l'ús de la llibreria *jBPT* 9.1.4 va resultar ser més complicat del que s'esperava. Aquesta eina no està ben documentada i fa temps que els desenvolupadors no hi introdueixen canvis ni millores. Això ha provocat que l'aprenentatge hagi sigut un procés llarg i complex, en el que ha calgut experimentar amb les classes i funcions de la llibreria per a entendre el seu funcionament. En aquesta experimentació també s'ha trobat que recórrer un arbre RPST 9.1.5 no és trivial, ja que, un dels mètodes que cal fer servir (obtenir els fills d'un node de l'arbre) no funciona correctament perquè retorna els fills desordenats. Això ha implicat crear funcions que resolguin aquest comportament erroni.

A causa dels obstacles esmentats anteriorment es va haver d'adaptar la planificació temporal. Es va canviar la data d'entrega i es va decidir dedicar menys hores setmanalment. També es va ampliar el nombre total d'hores dedicades a la tasca de llegir els models BPMN. Això va comportar que la tasca d'entrenar un model de *Deep Learning* amb les dades del *dataset* se suprimís de la planificació.

6.4.1 Canvis als objectius

Com s'ha comentat prèviament, els objectius van variar lleugerament, ja que en la planificació inicial estava previst crear un model predictiu, però aquest objectiu es va haver de suprimir. Es considera que no és una gran pèrdua perquè no era l'objectiu principal, sinó una forma d'ensenyar la feina feta. Tot i això, l'objectiu principal era generar un conjunt de dades i aquest s'ha mantingut en la planificació alternativa.

6.4.2 Canvis als costos

No hi ha hagut canvis als costos finals del treball. Malgrat que s'ha desenvolupat el treball amb més dies no hi ha hagut un augment de les hores de feina i, per tant, els costos de personal s'han mantingut. Tampoc s'han incrementat els costos per software perquè no es tenia cap llicència mensual.

7

Pressupost

En aquest apartat s'elabora una anàlisi econòmica del projecte amb la finalitat de decidir si és factible dur-lo a terme. Cal analitzar i estimar les següents classes de costos: costos de personal, costos de les tasques, maquinari, programari i costos d'espai de treball. També s'ha de tenir en compte els possibles contratemps que puguin sorgir i fer una estimació del que poden costar. A continuació es detallen les classes esmentades.

7.1 Costos de personal

Abans de res, s'ha de definir quins rols hi ha involucrats en el projecte. En aquest cas, hi ha el càrrec de Cap de Projecte, que és la persona encarregada de planificar, supervisar i dirigir-lo durant la realització d'aquest. És un rol compartit entre els codirectors del TFG i l'autor d'aquest. També existeix el rol de programador, que és la persona que dissenya, escriu i depura el codi del projecte. En aquest cas el rol només l'executa l'autor del TFG.

A la taula 7.1 es pot veure el sou brut, el que s'ha de pagar de Seguretat Social (SS) i la retribució total de cadascun dels càrrecs esmentats anteriorment. Els sous bruts han sigut extrets de la pàgina web Glassdoor [40].

Rol	**Sou brut**	**SS**	**Retribució**
Cap de projecte	22,5 €/hora	6,75 €/hora	29,25 €/hora
Programador	12 €/hora	3,6 €/hora	15,6 €/hora

Taula 7.1: Rols del projecte i la facturació per hora. Font: Elaboració pròpia.

7.2 Costos per tasca

El temps de desenvolupar cada tasca és diferent, per tant, cal calcular per a cada una de les activitats el seu cost. El temps de cada tasca està descrit a l'apartat de Planificació temporal 6. També cal considerar que cada tasca pot ser desenvolupada pel Cap del Projecte o pel Programador. Les activitats associades a la Gestió del Projecte (GP)

6.1.2 i a la Documentació 6.1.5 les durà a terme íntegrament el Cap del Projecte. Les tasques de Generació del Dataset 6.1.3 les desenvoluparà el Programador. En canvi, les tasques d'Entrenament del Model 6.1.4 les faran el Cap del Projecte i el Programador, això sí, el Programador és el que hi ha d'invertir més temps. En aquestes tasques el Cap del Projecte simplement ajudarà a dissenyar el model i decidirà si és correcte o no.

Codi	Nom Tasca	Cap del projecte	Programador	Cost
GP-C	Contextualització i abast	15 h	0 h	438,75 €
GP-T	Planificació temporal	13 h	0 h	380,25 €
GP-M	Metodologia	13 h	0 h	380,25 €
GP-P	Pressupost	13 h	0 h	380,25 €
GP-S	Informe de sostenibilitat	13 h	0 h	380,25 €
GP-I	Instal·lació programari	0 h	20 h	312,00 €
GD-O	Obtenir models BPMN	0 h	10 h	156,00 €
GD-P	Parsejar BPMN	0 h	14 h	218,40 €
GD-I	Identificar verbs	0 h	31 h	483,60 €
GD-D	Crear descripcions	0 h	47 h	733,20 €
GD-C	Crear dataset	0 h	28 h	436,80 €
MD-C	Carregar dataset	0 h	36 h	561,60 €
MD-E	Entrenar model	16 h	56 h	1.341,60 €
MD-P	Provar model	12 h	52 h	1.162,20 €
DO-M	Memòria	90 h	0 h	2632,50 €
DO-P	Presentació	30 h	0 h	877,50 €

Taula 7.2: Taula del cost per cada tasca. Font: Elaboració pròpia.

En la taula 7.2 s'hi troba el codi i nom de totes les tasques que s'han de realitzar. També s'hi veu el temps que hi dediquen tant el Cap del Projecte com el Programador. I finalment també hi ha el cost total de cada tasca, aquest s'ha calculat en funció del preu per hora calculat a la taula 7.1. En total el cost de totes les tasques és de **10.875,15 €**.

7.3 Costos genèrics

En aquest apartat s'expliquen els costos genèrics que té el projecte. Aquests costos són la suma dels costos de maquinari, de programari i del teletreball. A continuació es detallen cadascun d'ells.

7.3.1 Maquinari

En tot projecte informàtic es necessita un maquinari per a dur-lo a terme. En aquest cas s'ha fet servir un monitor *Newskill Icarus RGB IC27QRS 27"*, un portàtil *Acer Aspire 3 A315-51-59SU*, un ratolí *Logitech G603* i un teclat *Logitech G213*. A la taula 7.3 s'hi veu el seu preu de mercat l'octubre del 2021 i també el seu cost amortitzat. Per a calcular el cost amortitzat s'ha de tenir en compte que el maquinari té un temps de vida estimat

de 4 anys. Per a fer el càlcul de les hores reals que permet treballar cada eina cal tenir en compte que un any disposa d'uns 231 dies hàbils (251 dies laborables menys 20 dies de vacances) i que 1 dia hàbil compta de 8 hores. Com ja s'ha explicat a l'apartat de Planificació 6, es calcula que es farà servir aquestes eines unes 540 hores. Per tant, la fórmula que es fa servir és la següent.

$$4 \text{ anys} \cdot 231 \text{ dies laborables} \cdot 8 \text{ hores} = 7392 \text{ hores hàbils}$$

$$\frac{540 \text{ hores de projecte}}{7392 \text{ hores hàbils}} = 0,073$$

Per tant, es multiplica el preu de mercat de cada producte per $0,073$ per a obtenir el cost amortitzat.

Nom del maquinari	**Preu de mercat**	**Cost amortitzat**
Monitor	239 €	17,45 €
Portàtil	519 €	37,89 €
Ratolí	49 €	3,50 €
Teclat	62 €	4,53 €
Total	**869 €**	**63,37 €**

Taula 7.3: Costos del maquinari. Font: Elaboració pròpia.

7.3.2 Programari

Actualment, al mercat hi ha disponible una gran quantitat de programari. Aquest pot ser gratuït o de pagament. Normalment, es pot trobar software gratuït de qualitat que ens permet executar la tasca i a més estalviar diners. Per a dur a terme aquest treball s'ha intentat reduir al màxim els costos de programari. És per això que el llistat de programes que es mostra a la taula 7.4 és totalment gratuït. Alguns dels serveis que es faran servir també tenen una versió de pagament, però en principi amb la versió més bàsica d'aquests ens és suficient. També és important comentar que el *PyCharm Edu* és només gratuït per a estudiants, com que l'autor d'aquest treball ho és es pot fer servir, si no ens costaria 19,90 € al mes.

7.3.3 Teletreball

Com que el treball es realitza en mig d'una pandèmia, és millor evitar els espais públics i, per tant, el treball es fa íntegrament a casa. S'ha de calcular quant costa el fet d'estar teletreballant a casa. Això implica el mobiliari (taules, cadires, etc.), la llum, el lloguer i altres despeses. Alguns sindicats laborals [41] ja estan calculant quant costa el fet de treballar des de casa, i han arribat a la conclusió que aquestes despeses arriben a uns 21 € al mes. Per tant, si el projecte es desenvolupa en 7 mesos, el cost total és de **147 €**.

Nom programari	Cost mensual	Mesos utilitzats	Cost total
Python	0 €	6	0 €
PyCharm Edu	0 €	6	0 €
IntelliJ IDEA Community	0 €	6	0 €
Overleaf	0 €	7	0 €
Clockify	0 €	7	0 €
Github	0 €	7	0 €
Tensorflow	0 €	3	0 €
Google Colab	0 €	3	0 €

Taula 7.4: Costos pel programari. Font: Elaboració pròpia.

7.4 Total dels costos per personal i genèrics

En la taula 7.5 s'hi poden veure el total dels costos del personal i els costos genèrics calculats anteriorment.

Concepte	Cost
Personal	10.875,15 €
Maquinari	63,37 €
Programari	0,00 €
Teletreball	147,00 €
Total	**11.085,52 €**

Taula 7.5: Costos de personal i genèrics. Font: Elaboració pròpia.

7.4.1 Contingència

Cal tenir present que durant el projecte surtin dificultats i adversitats. Aquestes poden fer que el cost final del projecte s'encareixi. Per tant, cal estar preparats per si passa. Es prepara una partida de contingència, o marge de seguretat, que permetrà cobrir els imprevistos no anticipats. Es calcula com un percentatge del total del pressupost.

En els projectes de desenvolupament software s'acostuma a reservar entre un 10% i un 20% del pressupost. Com que es té un nivell de detall del pressupost òptim, només s'hi destinarà el 10%. Per tant, el cost total de la contingència es calcula així:

$$\text{Cost total} \cdot \ 10\% = 11.085,52 \cdot \ 10\% = \mathbf{1.108,55\ €}$$

7.4.2 Imprevistos

En darrer terme també s'ha de considerar les activitats que formen part dels plans alternatius. Aquestes activitats no es costegen totalment, sinó que es fa segons el percentatge

estimat de risc que tinguin cadascuna d'elles. A la taula 7.6 es veuen els possibles imprevistos que poden sorgir, l'increment de cost que suposarien i el cost que finalment es reserva.

Imprevist	**Cost**	**Risc**	**Total**
Nou ordinador	500,00 €	5%	25,00 €
Increment del temps d'entrenament (20h)	339,30 €	20%	67,86 €
Google Collab Pro (5 mesos)	49,95 €	10%	4,99 €
Total	**889,25 €**		**97,86 €**

Taula 7.6: Taula amb els costos dels imprevistos. Font: Elaboració pròpia.

7.5 Cost total del projecte

En la taula 7.7 s'hi calcula el cost total d'aquest.

Concepte	**Cost**
Tasques	10.875,15 €
Maquinari	63,37 €
Programari	0,00 €
Teletreball	147,00 €
Contingència	1.108,55 €
Imprevistos	97,86 €
Total	**12.291,93 €**

Taula 7.7: Taula amb els costos totals del projecte. Font: Elaboració pròpia.

7.6 Control de gestió

Per a evitar sorpreses en el cost s'ha de realitzar un control dels costos. Es tracta d'avaluar les possibles desviacions que hi hagi durant el transcurs del projecte. Es comparen els costos reals que s'hagin tingut al treball amb el pressupostat. En cas de detectar una desviació cal saber en quines activitats o etapes ha ocorregut, explicar el perquè ha passat i quina quantitat.

En cas que es produeixin desviacions en tasques mencionades a l'apartat d'imprevistos 7.4.2 s'utilitzarà la partida destinada a cobrir aquests contratemps. Si no fos suficient o les desviacions fossin d'una tasca que no s'ha tingut en compte s'hauria de fer servir el fons de contingència. Les fórmules que s'utilitzen per a fer el seguiment són les següents:

$$\text{cost estimat} - \text{cost real}$$

$$\text{hores estimades} - \text{hores reals}$$

8

Informe de sostenibilitat

Un cop respost el qüestionari del projecte EDINSOST [42] sóc més conscient del grau de coneixement que tinc respecte a la sostenibilitat. Actualment, es posa molt èmfasi en la sostenibilitat, el canvi climàtic i el medi ambient, això fa que es tinguin coneixements teòrics generals sobre aquests temes. Malgrat tot, els coneixements pràctics necessaris per a acomplir un projecte de forma sostenible no els tinc.

Això implica que a l'hora de detectar que un element del treball no és sostenible ho puc fer amb certa facilitat. Però, no tinc prou coneixements per a mesurar-ho amb els indicadors adequats. També es fa difícil proposar alternatives més sostenibles, ja que ignoro les tècniques que es fan servir actualment.

Pel que fa a l'impacte social que un servei TIC pot tenir em considero capacitat per a avaluar el paper que juga aquest projecte en la societat. Tant en l'àmbit de l'accessibilitat, i la qualitat ergonòmica com per la justícia social, l'equitat i la diversitat. En aquest cas també em considero capaç de proposar mesures eficaces per a millorar en aquests àmbits. Encara que també costa fer servir indicadors per a estimar com un projecte contribueix a millorar el bé comú de la societat.

Respecte a la dimensió econòmica no comprenc gaire bé les parts econòmiques d'un projecte, ja que és un àmbit molt extens, però sí que en tinc alguna idea.

I per finalitzar, considero que sé valorar els beneficis del treball en equip en les TIC, i a més, conec les eines necessàries per a desenvolupar projectes amb més gent.

8.1 Dimensió econòmica

En l'apartat de Pressupost 7 s'ha detallat el cost del projecte i dels seus apartats per a tenir clar quines són les seccions amb una partida pressupostària més alta. Com s'ha explicat, s'ha intentat que tingui un cost mínim. Això implica fer servir programari gratuït i utilitzar maquinari que no sigui de gamma màxima.

El punt on més diners es destinen és en els costos personals. Evidentment, aquests no es poden reduir més, ja que el sou dels treballadors ha de ser just i les hores que han de fer ja estan marcades per les característiques del TFG. En conseqüència, s'ha aconseguit que el cost estimat per a la realització del projecte sigui adequat.

Pel que fa a les solucions actuals, el projecte també pretén millorar-les en l'aspecte econòmic. Aquesta solució permetrà estalviar diners als usuaris d'aquest servei, ja que no es necessitarà tant personal qualificat per a realitzar els models BPMN.

8.2 Dimensió ambiental

La dimensió ambiental de la realització del projecte s'estima que serà baixa. Això es deu al fet que no s'utilitza maquinari que consumeixi grans quantitats d'energia. A més, el treball es duu a terme plenament des de casa i, per tant, no es fan desplaçaments innecessaris que podrien tenir un cost ambiental elevat. Tampoc es produeixen residus, ja que el poc paper que es fa servir és o bé reciclat o bé reutilitzat.

Respecte a la seva vida útil aconsegueix reduir l'impacte que tenen les alternatives actuals. Amb l'ús de la solució plantejada en aquest projecte es redueix l'impacte ambiental, ja que es podrà reduir personal que treballi a fer models manualment. Això implica una reducció en desplaçaments, en espai de treball i en llum a les oficines.

8.3 Dimensió social

En l'àmbit personal crec que el projecte m'aportarà coneixements tècnics en el sector del Deep Learning, coneixements sobre realitzar correctament un projecte d'enginyeria informàtica, capacitat per a entendre què són els models de processos empresarials, aprendre a gestionar millor les tasques, millorar la capacitat per a executar projectes a temps, també em proporcionarà experiència a l'hora de programar.

La utilitat d'aquest treball a nivell col·lectiu es basa en el fet que les empreses que facin servir BPMN podran reduir la quantitat de feina que els *Business Analysts* han de fer, i, per tant, podran dedicar més temps a altres coses més necessàries. També pot ser que a la llarga es requereixin menys treballadors en una empresa fent aquesta feina, per a la persona que es dedica a això pot ser un problema, però al final la tecnologia sempre ha implicat canvis en totes les professions.

9

Solució plantejada

En aquest apartat es detalla com s'ha implementat la solució al problema. A més, s'hi comenten els canvis que s'han hagut de fer respecte a la planificació inicial, i els obstacles que han obligat a fer aquestes modificacions.

Com que aquest projecte compta de diverses etapes, s'ha decidit desenvolupar cada etapa com a un petit programa que sigui independent dels altres. D'aquesta manera s'aconsegueix que el codi sigui més modular, per tant, tal com s'explica en la web *Software Design Principles* [43], el codi és més fàcil d'ampliar, provar, corregir errors i reutilitzar.

9.1 Eines utilitzades

Tal com afirma Madhuri Hammad [44] en la web *Geeks for Geeks*, és important no reinventar la roda quan es desenvolupa un projecte de *Software*. Això vol dir que no cal programar de zero una eina, se'n pot fer servir una que ja ha estat inventada.

En aquest treball s'ha decidit fer servir un conjunt d'eines que permeten agilitzar algunes fases del treball i no malbaratar temps i recursos. A continuació, s'expliquen les eines que s'han fet servir.

9.1.1 Camunda

Camunda Platform [45] és un sistema de flux de treball i de suport a decisions de codi obert. Aquesta plataforma conté eines per a crear fluxos de treball i models de decisió. També permet als usuaris executar les tasques dels fluxos de treball [46]. Ells mateixos expliquen que el seu objectiu és permetre a empreses dissenyar, automatitzar i millorar els seus processos empresarials. Ha estat desenvolupat en Java i es pot fer servir en qualsevol sistema operatiu que admeti aquest llenguatge.

Figura 9.1: Logotip de Camunda. Font: camunda.com

Tal com explica Charles Humble [47] aquesta plataforma té un motor de processament de models BPMN 2.0 implementat en Java. Consta també de *Cockpit*, una eina de seguiment i administració. Un *plugin* per a *Eclipse* i un producte anomenat *Cycle* que es fa servir per sincronitzar diagrames BPMN amb els executables desenvolupats amb *Modeler*.

Com es pot veure, *Camunda Platform* és una eina molt completa, tanmateix, en aquest projecte només es fa servir com a *parser* dels models BPMN. És a dir, l'eina s'utilitza per a llegir els diagrames emmagatzemats en XML i convertir-los en objectes de Java. Aquests contenen mètodes de consulta que permeten extreure els atributs que es consideren més importants i després guardar-los en una altra estructura de dades. En aquest cas, es guarden en un fitxer JSON.

9.1.2 Freeling

Tal com s'ha explicat en l'apartat Abast 4 en aquest treball també es fa servir *FreeLing* [28] [29] [30] [31] [32]. És un projecte de codi obert dirigit per Lluís Padró, un dels directors d'aquest treball. Es va crear amb l'objectiu de compartir amb la comunitat els resultats de la recerca realitzada pel grup de recerca en processament del llenguatge natural de la UPC. *FreeLing* es desenvolupa i es manté pel Centre de Recerca TALP [48], a la Universitat Politècnica de Catalunya. També rep moltes contribucions externes de part de la comunitat.

És una llibreria de *C++* que permet l'anàlisi del llenguatge en molts idiomes. Alguns d'aquests idiomes són l'anglès, el català, el portuguès, el rus, l'alemany, el castellà, entre d'altres. Alguns dels principals serveis que inclou aquesta llibreria són la *tokenització* de text, divisió de frases, una anàlisi morfològica, reconeixement de paraules compostes, etiquetatge PoS, etc.

En aquest projecte es fa servir una versió modificada del *Freeling* que utilitza una gramàtica per a buscar frases en forma d'accions, com per exemple "enviar missatge al client". Aquesta adaptació retorna un JSON amb informació sobre el predicat, l'objecte directe i altres complements de la frase. També retorna informació sobre el gènere,

el número, el temps del nucli de cada component de l'acció.

9.1.3 SimpleNLG

SimpleNLG [23] és una API de Java per a generar textos en llenguatge natural. El codi original va ser desenvolupat per Ehub Reiter al departament de ciències de la computació de la universitat d'Aberdeen, però com que és un projecte de codi obert, ha tingut diverses contribucions per part de la comunitat. També existeix una versió de l'API per a Python, desenvolupada per Brad Jascob [49]. Aquesta adaptació és pràcticament igual a l'original, malgrat tot, no totes les classes i els mètodes han pogut ser replicats.

Aquesta eina implementa un sistema lèxic i morfològic que calcula les formes flexionades, un realitzador que genera textos de forma sintàctica, i un microplanificador. Originalment, es va desenvolupar per a crear textos en anglès, avui en dia ja funciona amb el francès, l'italià, el brasiler, el neerlandès, l'alemany o el gallec.

En aquest projecte es fa servir la funcionalitat del *realiser* o realitzador. Aquesta permet crear frases a partir de crides a funcions. Les frases es poden crear mitjançant un text o *string*, o bé, definint per separat els components d'una frase (subjecte, complements, predicat, etc.). Aquesta flexibilitat és molt pràctica perquè permet crear una frase de *SimpleNLG* de la manera més convenient.

Amb les dades obtingudes prèviament pel *Freeling* 9.1.2 es pot generar una frase nova. La utilitat d'això és que es pot ajuntar la persona que realitza l'acció (*lane*) amb la mateixa acció (text de l'element BPMN) i crear una frase amb sentit. Després, aquestes frases es concatenaran, fent servir les *CoordinatedPhrase* i els *DocumentElement*, per a formar paràgrafs que descriuran el procés BPMN.

9.1.4 jBPT

La llibreria de Java *jBPT* [50] és un conjunt de tecnologies que ajuden a la investigació sobre disseny, execució i avaluació de processos empresarials. Aquesta llibreria ofereix una àmplia gamma d'anàlisis i utilitats que es poden ampliar fàcilment perquè és de codi obert. Vas ser desenvolupada l'any 2012 per Artem Polyvyanyy i Matthias Weidlich entre molts d'altres.

Algunes de les seves funcionalitats són les tècniques per calcular RPST, arbres de flux de treball, descomposició modular d'un arbre, desenvolupar i desenredar d'un sistema de xarxa de Petri, entre moltes altres.

En el projecte es fa servir per a guardar el model BPMN com a un graf i posteriorment convertir-lo a un *Refined Process Structure Tree (RPST)* 9.1.5. Això permet llegir el BPMN com un arbre i, per tant, evitar cicles. En l'última part del treball això és molt útil de cara a ajuntar les frases en un sol paràgraf.

9.1.5 RPST

Un RPST (Refined Program Structure Tree) és un arbre que s'utilitza per a *parsejar* models BPMN, aquests contenen en els seus nodes la informació necessària per a emmagatzemar els elements d'un model i les relacions entre ells. Formalment, es pot descriure de la següent manera.

Un RPST és l'arbre dels fragments canònics d'un model de procés G, de manera que el pare d'un fragment canònic F és el fragment canònic més petit de G que conté F [51].

Aquests arbres van ser creats per Jussi Vanhatalo, Hagen Völzer i Jana Koehler amb la intenció de millorar les tècniques de *parseig* de BPMN que existien. La seva solució aconsegueix una descomposició més fina. L'arbre resultant de *parsejar* un BPMN amb aquesta tècnica és únic, modular i granular. A més, es pot calcular en temps lineal.

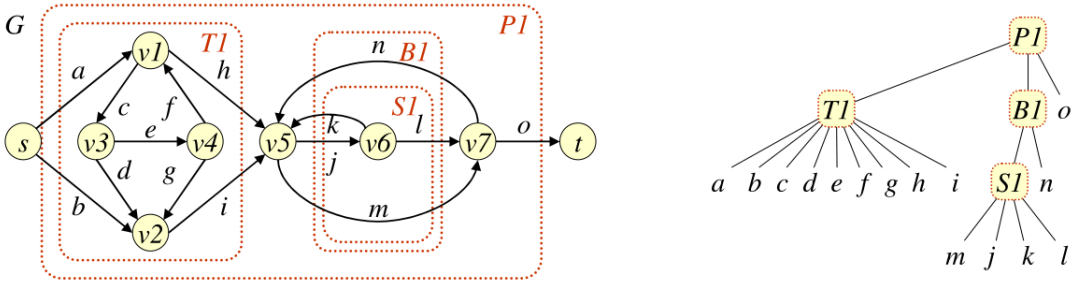

Figura 9.2: Exemple de conversió d'un graf BPMN a un arbre RPST. Font: The Refined Process Structure Tree. Jussi Vanhatalo, Hagen Völzer, Jana Koehler

Els nodes d'aquests arbres s'anomenen fragments. Cada fragment representa una secció del model *parsejat*. Aquesta secció té un element d'entrada i un de sortida. Vol dir que per arribar a tots els elements BPMN que són dins del fragment cal haver passat abans per l'element d'entrada, i tots els elements de dins el fragment han d'arribar a l'element de sortida. Cada fragment de l'arbre pot tenir més fragments com a fills. Això implica que els RPST donen una forma ordenada de poder recórrer el model.

En un RPST hi ha diferents tipus de fragments [51].

- **Trivials**: Són fragments que contenen exactament una aresta.
- **Polígon**: Un polígon és un graf que conté almenys tres nodes, exactament tantes arestes com nodes, de manera que hi ha un cicle que conté tots els seus nodes i totes les seves arestes. En la figura 9.3 els grafs $P1$ i $P2$ ho són.
- **Bond** o Vincles: Són grafs que conté exactament dos nodes i almenys dues arestes entre ells. En la figura 9.3 els grafs $B1$ i $B2$ ho són.

- **Rígid**: Són fragments que no són ni un component trivial, ni polígon, ni enllaç. Aquests són els més difícils de treballar, ja que per llegir els elements no hi ha un ordre clar establert. En aquest treball s'ha decidit no tractar-los perquè no hi ha prou temps de crear una estratègia per processar-los. En la figura 9.3 els grafs $T1$ i $T2$ ho són.

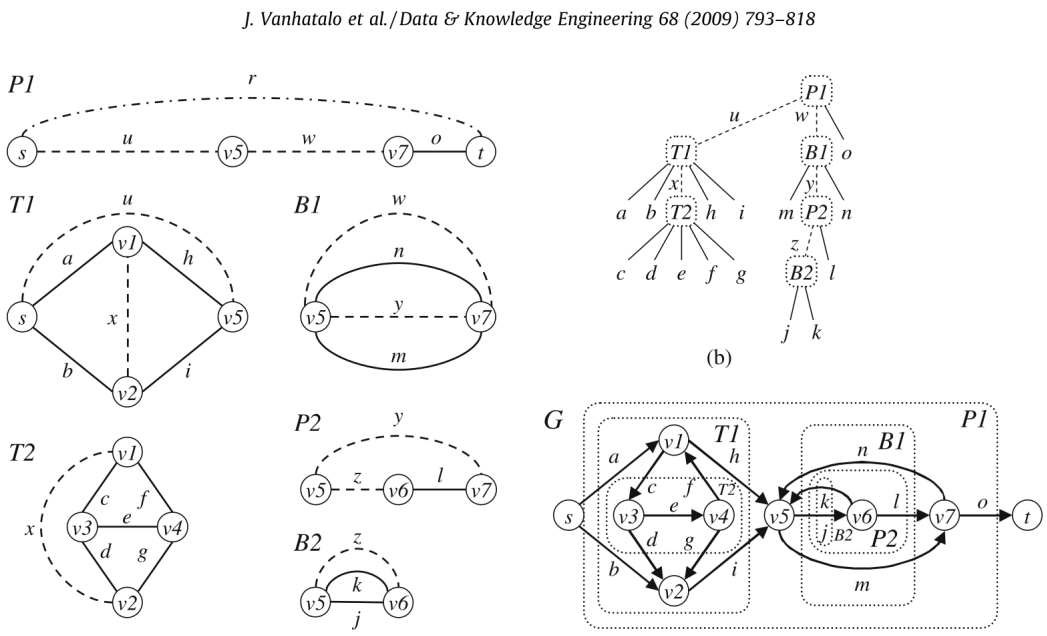

Figura 9.3: Tipus de fragments d'un RPST. Font: J. Vanhatalo et al. Data & Knowledge Engineering 68 (2009) 793–818

9.2 Anàlisi d'alternatives

En aquest apartat s'expliquen les decisions que s'han pres i el motiu de cada una d'elles.

9.2.1 Parser BPMN

Inicialment, es va pensar que seria millor desenvolupar el projecte en Python, ja que és més fàcil de fer servir i l'autor està més acostumat a utilitzar-lo. Per això es va començar a buscar *parsers* de BPMN que es poguessin fer servir en aquest llenguatge. Desafortunadament, no existeixen llibreries fiables i es va haver d'optar per un projecte de codi obert anomenat *python-bpmn-engine* [52]. Aquest no comptava amb una bona documentació i no feia exactament el què es volia i finalment es va descartar usar-lo.

Amb els directors del projecte es va proposar implementar un *parser* des de zero en Python, o bé fer servir la llibreria *Camunda*. Fer-lo de nou tenia els avantatges de ser en un llenguatge més tractat pel desenvolupador i que es podia dissenyar com es volgués. Per contra, implicava moltes hores de feina per a tenir una versió final. Finalment, després d'aconseguir instal·lar el *Camunda* es va fer servir aquesta llibreria. Està molt ben documentada i ja té molts mètodes creats. Per tant, els únics problemes van ser a l'hora d'instal·lar-la i d'aprendre el seu funcionament.

9.2.2 FreeLing

La llibreria *FreeLing* es va escollir perquè un dels directors va participar en el seu desenvolupament. Això implica que durant el treball s'ha tingut suport de primera mà per part d'un dels creadors. Ha sigut especialment útil quan s'ha trobat algun problema amb l'eina perquè ha sigut molt senzill de resoldre.

9.2.3 SimpleNLG i jBPT

En el cas d'aquestes dos llibreries també van ser propostes dels directors del projecte. Tot i que també s'ha estudiat si existeixen alternatives millors, no se n'ha trobat. En el cas de *SimpleNLG* també es va analitzar si era millor fer servir la llibreria en Python o Java. Al final es va decidir en Java, ja que era el llenguatge original i estava plenament documentat i desenvolupat.

9.3 Implementació

En aquesta secció s'explica com s'ha desenvolupat cada part del projecte. El desenvolupament del projecte s'ha dividit en cinc parts diferents. Cada mòdul és independent de la resta, això vol dir que sempre que es disposi d'una entrada adequada el mòdul pot executar-se sense executar la resta. Això permet una gran flexibilitat tant a l'usuari del programa, com als desenvolupadors. Tots els mòduls llegeixen els fitxers corresponents (el *path* s'indica abans d'iniciar el programa), els tracten i generen nous fitxers. Aquests fitxers de sortida són els que, més tard, es poden utilitzar com a entrada de la resta de mòduls.

Malgrat ser independents, la primera execució (quan encara no s'ha obtingut cap fitxer de sortida) cal fer-la seqüencialment i en el mateix ordre que estan descrits a continuació.

En la figura 9.4 s'observa el conjunt de mòduls que conformen el projecte i com es relacionen entre ells. Una aresta entrant indica l'extensió dels fitxers d'entrada, i una aresta sortint indica l'extensió dels arxius de sortida. Si hi ha un asterisc davant del nom de l'extensió, vol dir que aquell mòdul llegeix o escriu múltiples arxius. També s'hi indica el directori on s'emmagatzema cada arxiu per defecte.

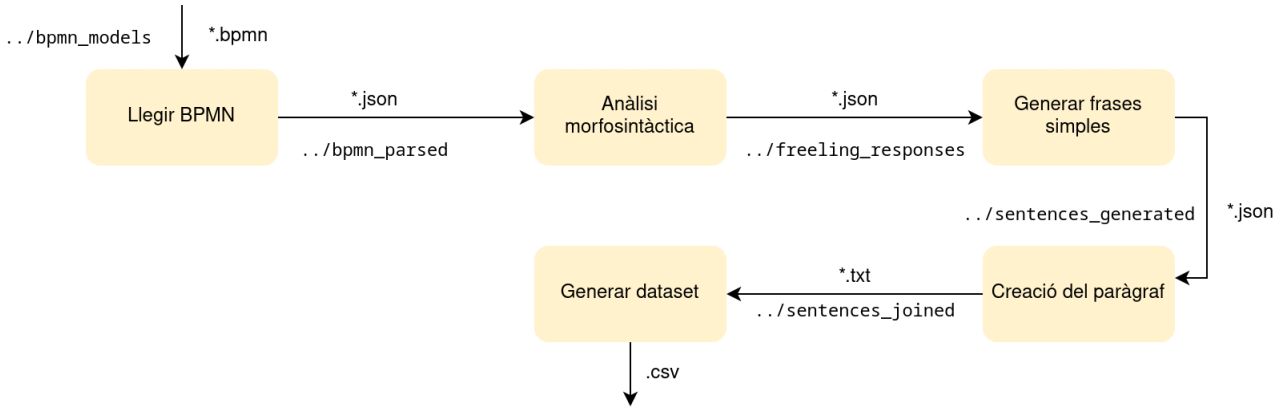

Figura 9.4: Diagrama que mostra la relació entre els mòduls del projecte. Font: Elaboració pròpia

9.3.1 Llegir els models BPMN

Aquest mòdul és l'encarregat de llegir (*parsejar*) els models BPMN i obtenir la informació necessària per a poder tractar-los més tard.

El programa busca per defecte els models a la carpeta `../bpmn_models`, però es pot canviar passant un argument amb el *path* desitjat a l'executable del programa.

Dins d'aquesta carpeta s'hi busquen tots els arxius `.bpmn` que hi hagi (encara que estiguin en subcarpetes). Aquests arxius són XML modificats que serveixen per a especificar els diagrames.

Per a llegir aquests fitxers s'ha fet servir la llibreria *Camunda*. Tal com s'ha explicat en l'apartat 9.1.1, la llibreria ens serveix per a convertir l'entrada en objectes de *Java*. D'aquesta manera es poden llegir i tractar els elements que es consideren necessaris. Aquests elements es guarden dins de la carpeta `../bpmn_parsed` en un fitxer JSON.

En un mateix model BPMN pot haver-hi varis *Start Events* (esdeveniments inicials). Això pot passar quan hi ha diversos subprocessos dins del model. Això suposava un problema a l'hora de crear l'arbre RPST en el mòdul d'unir les frases 9.3.4. Per evitar dificultats s'ha decidit tractar cada subprocés com si fos un model BPMN diferent.

Per a cada arxiu es guarden tots els seus elements (esdeveniments, activitats i portes). I per cada element s'emmagatzema l'identificador, el nom, el tipus, el *lane* al qual pertany i l'identificador i nom de l'aresta de cada element que segueix. En la figura 9.5 s'hi veu un exemple d'un BPMN *parsejat*.

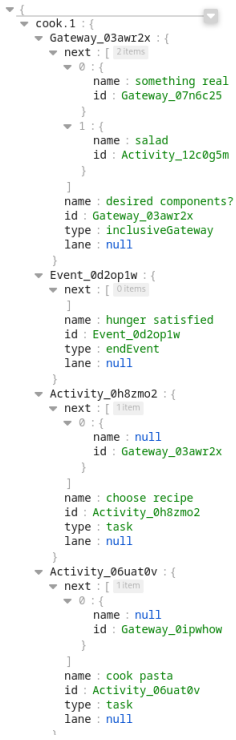

Figura 9.5: Part d'un BPMN *parsejat* guardat com a JSON. Visualitzat en la web JSON Editor. Font: Elaboració pròpia

9.3.2 Anàlisi morfosintàctica

Per a obtenir l'anàlisi morfosintàctica dels noms de cada element d'un BPMN, s'ha creat un client de *Freeling* 9.1.2 en *Python*. És imprescindible que per a córrer aquest programa estigui executant-se una imatge personalitzada del *Docker* del *Freeling*. Aquesta imatge ha estat adaptada pels directors del projecte per a poder-la fer servir en aquest projecte de fi de grau.

El client està configurat amb uns paràmetres per defecte que permeten carregar els arxius d'entrada, la connexió amb el *Docker* i escriure la sortida. Els paràmetres i els seus valors per defecte són aquests:

- `-l` o `--lang`: Configura l'idioma del *Freeling*. Per defecte `en`.
- `-p` o `--parsedpath`: *Path* dels JSON d'entrada. Per defecte `../bpmn_parsed`.
- `-dh` o `--host`: *Host* de la imatge de *Freeling*. Per defecte `172.17.0.2`.
- `-lp` o `--langport`: *Port* del detector d'idioma de *Freeling*. Per defecte `50005`.
- `-fp` o `--freelingport`: *Port* del *Freeling*. Per defecte `60006`.

- `-r` o `--resppath`: *Path* dels JSON de sortida. Per defecte `../freeling_responses/`.

L'entrada que rep el client és el conjunt de fitxers JSON que s'ha creat en el mòdul anterior. Un cop llegit un fitxer, es processa i s'envia el nom de cada element a l'analitzador de frases. Un cop analitzat, es rep un objecte JSON. Es prepara la informació retornada i s'afegeix en el JSON original per a formar un arxiu nou. Aquests arxius són la sortida del programa.

En el cas que alguna frase no segueixi el format acceptat pel *Freeling* (només accepta accions) pot ser que no es retorni res. Llavors, no es pot aprofitar tot el potencial del *SimpleNLG*, ja que aquesta eina necessita que li indiquis clarament quins són els elements de la frase. És a dir, cal indicar el verb, els subjecte, els adjectius, etc.

En aquests casos es pot obviar l'anàlisi o bé tractar l'element abans d'enviar-lo. Obviar l'anàlisi implica crear un objecte de *SimpleNLG* com si fos un *string*, per tant, no és recomanable.

És millor intentar fer un pretractament. Però és complex, ja que no sempre cal actuar igual. Un dels problemes que té *Freeling* és detectant frases que contenen un signe d'interrogació. Per això, el que es fa és suprimir l'interrogant abans d'enviar-los al *Freeling*. Actualment, aquest és l'únic pretractament que reben les frases abans de ser processades pel *Freeling*.

9.3.3 Generar frases simples

En aquest mòdul s'agafen els noms de cada element i es processen amb *SimpleNLG* 9.1.3 fent servir la informació recopilada en el bloc anterior. Es tracta de preparar els noms dels elements perquè tinguin un aspecte més natural, en comptes del format d'acció (verb + objecte). Hi ha frases que han de ser dutes a terme per una persona o entitat, en BPMN es defineix com a *lane*. Sempre que un element en tingui, aquest serà afegit com a subjecte de la frase.

L'entrada del programa són els arxius `.json` que s'han creat anteriorment. Per defecte, es busquen en la carpeta `../freeling_responses/`, però aquest *path* es pot canviar enviant un argument a l'executable amb la ruta desitjada.

Aquest mòdul està programat en *Java*, ja que la versió de *SimpleNLG* era més fiable i estable en aquest llenguatge.

Un cop s'ha processat cada frase d'un model, es modifica el JSON original per a contenir la informació nova. Aquesta és un *string* `finalSentence` que conté la frase final generada. I també un objecte de JSON `finalPhrase`, que conté els elements necessaris per a poder reconstruir una frase en *SimpleNLG*. Això ens serà útil en el mòdul següent per a poder utilitzar les noves frases creades.

Aquesta informació es guarda en la ruta `../sentences_generated/`. En la figura 9.6 s'hi pot observar un exemple d'un fitxer JSON generat per aquest programa.

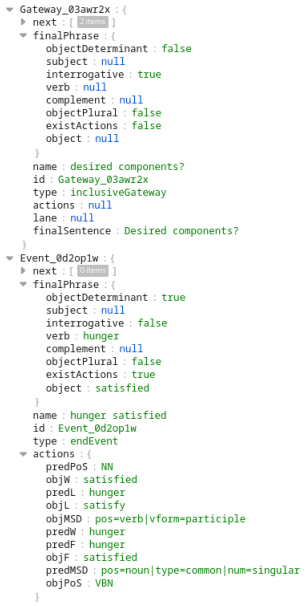

Figura 9.6: JSON modificat pel mòdul *Sentences Generator*. Visualitzat en la web JSON Editor. Font: Elaboració pròpia

9.3.4 Crear el paràgraf

Aquest mòdul és el més important de tot el projecte. En ell s'hi processen les frases simples que s'han generat anteriorment i s'uneixen per a crear el paràgraf. Aquest és la descripció final del model BPMN. La connexió entre dos frases o més no és trivial. Per a cada unió cal afegir-hi un petit text que expliqui o doni context sobre les frases que s'estan agrupant. Per a fer-ho cal tenir en compte quins tipus d'elements s'està ajuntant.

Distingim diversos tipus d'elements d'un BPMN a l'hora d'unir-los. I això ens permet afegir-hi el fragment de text adequat en cada cas. Els diversos casos que es tracten són:

- **Inicial**: Són els primers esdeveniments que ocorren. Davant de la frase pròpia de l'element s'hi poden afegir els següents connectors. *Once*, *First of all*, *When*, *The process starts when*, *Firstly*.
- **Enumeració d'activitats**: Són activitats o esdeveniments que ocorren en sèrie, sense cap bifurcació entre elles. S'ha decidit separar-les per comes, excepte l'última que té una conjunció al davant. El grup de conjuncions que es poden fer servir són els següents. *Then*, *After*, *And*, *Later*, *Next*, *After that*.
- **Bifurcació** o **Gateway**: Són els esdeveniments on el flux es divideix. La frase de la bifurcació s'envolta del següent text. *The condition [GATEWAY] is checked*
- **Branques de la bifurcació**: Són el conjunt d'esdeveniments que passen després que el flux s'hagi dividit, és a dir, després d'una bifurcació. S'hi afegeix la

següent frase que explica quina condició de la *gateway* cal complir perquè la branca s'executi. *If the answer is [CONDITION] then [BRANCH]*

Com es pot veure s'ha afegit diversos connectors per a cada tipus de frase a tractar. Això s'ha fet per a afegir diversitat al *dataset* final, ja que s'evita que totes les descripcions resultants continguin les mateixes construccions. El programa s'ha dissenyat perquè es pugui escollir manualment el text desitjat, o bé, se seleccioni aleatòriament.

L'entrada del programa són els arxius `.json` que s'han generat en el mòdul anterior. Per defecte, es busquen en la carpeta `../sentences_generated/`, però aquest *path* es pot canviar enviant un argument a l'executable amb la ruta desitjada.

Aquest programa s'ha desenvolupat fent servir el llenguatge de programació *Java*. S'ha decidit aquest per a poder fer ús de les llibreries *jBPT* 9.1.4 i *SimpleNLG* 9.1.3. La llibreria *jBPT* es fa servir per a generar un arbre RPST a partir del model i poder-lo recórrer. I la llibreria *SimpleNLG* es fa servir per a fer que la unió entre les frases sigui més natural.

Un cop obtingut l'arbre RPST es recorre amb un mètode recursiu que va des de baix cap a dalt. Això vol dir que les fulles són les primeres a ser tractades. La frase de l'element de la fulla es fa servir per inicialitzar un objecte de la classe *Sentence*. Aquesta classe transforma la frase de l'element BPMN i la converteix en una instància de *SimpleNLG*, això permet que posteriorment es pugui combinar aquesta frase amb altres segons convingui.

La figura 9.7 mostra les classes que s'han creat per a desenvolupar aquest mòdul. En el diagrama UML s'hi pot veure com estan relacionades les classes entre elles. I també s'hi mostren els atributs i mètodes públics més importants de cadascuna d'elles.

En la figura 9.8 hi podem veure els fragments de l'arbre RPST del model BPMN Cook. Els fragments $P1$, $P2$, $P3$, $P4$ i $P5$ són polígons. Els fragments $B1$ i $B2$ són *bonds*. I els fragments a, b, c, d, e, f, g, h són trivials, és a dir, són arestes del model.

En la figura 9.9 tenim una representació de l'arbre RPST. En aquest cas, el node arrel és el $P1$. Aquest és el node que conté tots els elements del model. El RPST es travessa començant per aquest node i se segueix pels seus fills. Un node no es processa fins que tots els seus fills han sigut processats anteriorment.

Un cop obtinguts tots els paràgrafs es guarden amb format `.txt/`. Les descripcions resultants s'emmagatzemen per defecte en la ruta `../sentences_joined/`, però aquest *path* es pot canviar enviant un segon argument a l'executable amb la ruta desitjada.

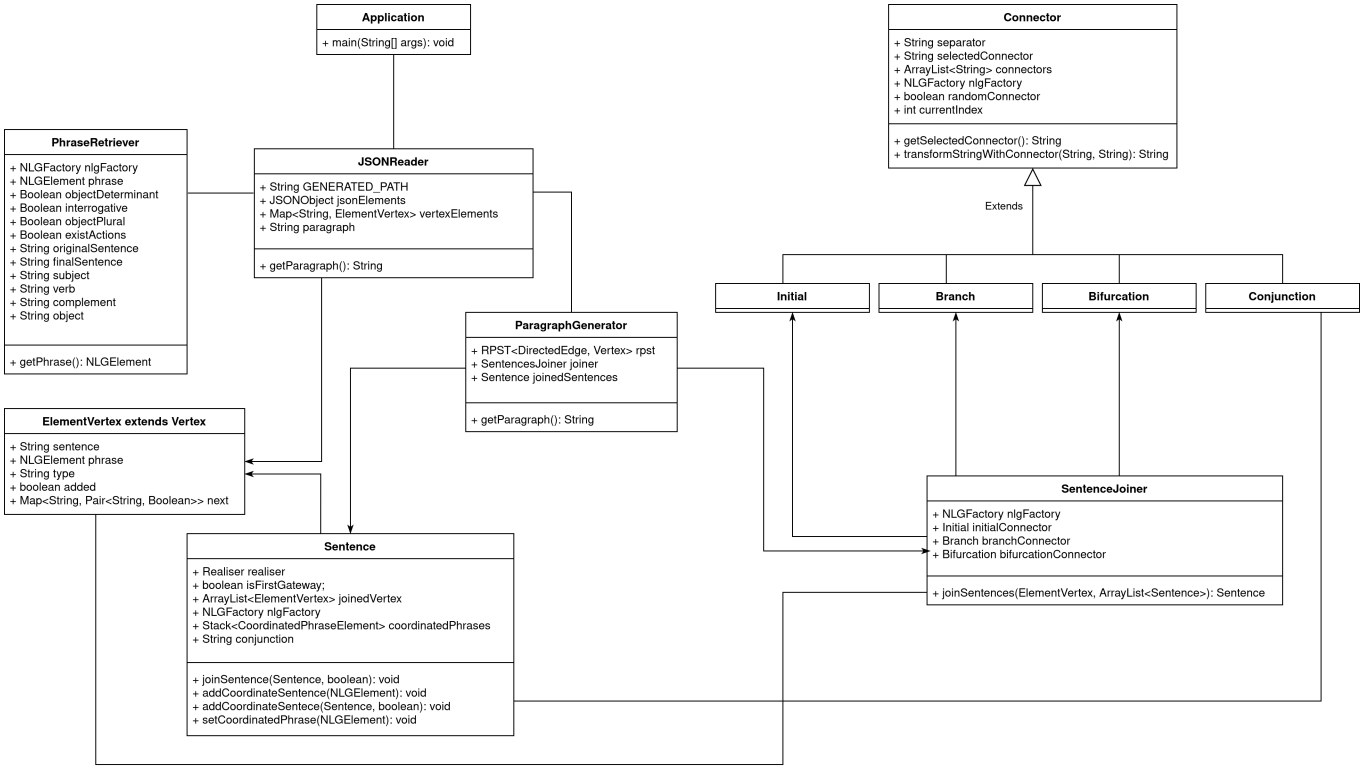

Figura 9.7: Diagrama de classes pròpies utilitzades en crear el paràgraf. Font: Elaboració pròpia

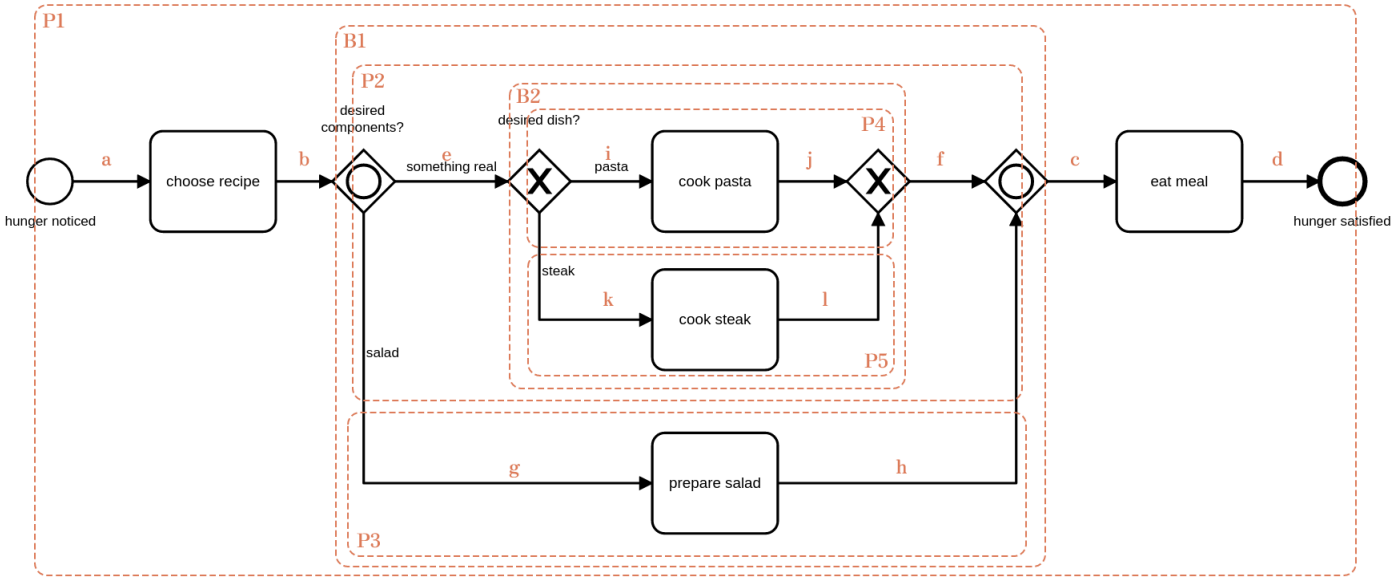

Figura 9.8: Fragments de l'arbre RPST del model d'exemple Cook. Font: Elaboració pròpia

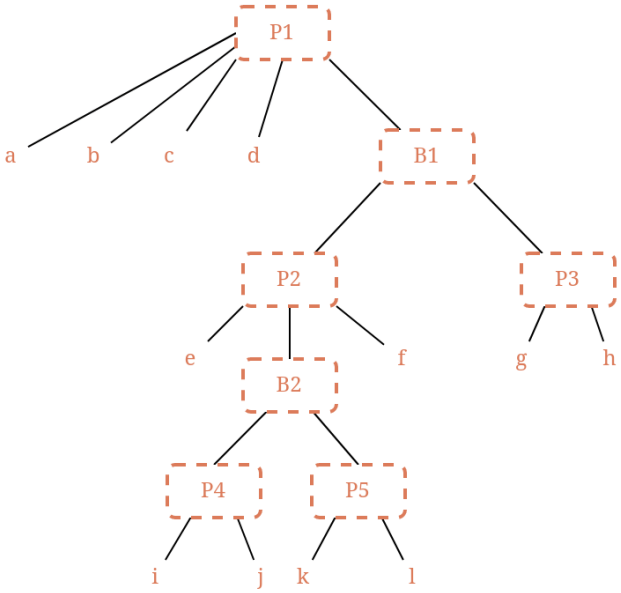

Figura 9.9: Arbre RPST del model d'exemple Cook. Font: Elaboració pròpia

10

Anàlisis del Dataset

En aquest apartat s'hi fa una breu descripció dels models utilitzats per generar el *dataset*, una explicació del contingut del conjunt de dades i finalment, també s'hi mostra una anàlisi de les descripcions generades.

10.1 Models BPMN utilitzats

S'han fet servir un total de 29 models. Per a obtenir un *dataset* divers, s'ha procurat que els models siguin prou diferents entre ells. Els models es poden categoritzar en molt simples (no tenen bifurcacions), no gaire complexos (tenen poques bifurcacions, no gaires elements i pocs o cap *lane*) i complexos (tenen moltes bifurcacions, bastants elements i diversos *lane*).

Cinc d'ells contenen rígids, que com s'ha comentat anteriorment 9.1.5 no es tracten en aquest projecte, però s'ha considerat important mantenir-los per a saber què genera el programa. En un *dataset* que es vulgui utilitzar per entrenar un model d'aprenentatge automàtic s'haurien d'ometre.

També n'hi ha que representen un procés real, però alguns d'ells són models BPMN d'exemple que no són processos de negoci reals. Aquests últims, encara que siguin mostres, serveixen per a comprovar que el programa llegeix les frases en l'ordre apropiat, i que els connectors s'afegeixen correctament. En cas d'utilitzar-los per a entrenar un model, s'hauria de discutir si cal afegir-los, ja que, potser són útils.

Els arxius BPMN han sigut extrets de diverses fonts. 15 models provenen del repositori de *Github* creat pel BPMN Model Interchange Working Group [53]. També s'han fet servir 9 models del concurs Process Model Matching Contest [54]. La resta són de la web de Camunda [46] o creats manualment.

10.2 Contingut de dataset

El conjunt de dades final conté una carpeta per a cada model generat. Dins de cada carpeta s'hi troben els següents arxius:

- Un únic arxiu amb extensió `.bpmn`.
- Un o diverses imatges `.png` que mostren el model. Les carpetes que contenen més d'una imatge mostren el contingut dels subprocessos, però en arxius separats per a facilitar la lectura.
- Un o diversos arxius de text amb extensió `.txt`. Aquests arxius són les descripcions dels models generats pel programa. Cada model té tantes descripcions com *start events* té. És a dir, les descripcions dels subprocessos es guarden en un fitxer a part.

10.3 Anàlisi de les descripcions

11

Treball futur

Un cop acabat el projecte, cal pensar quins han de ser els següents passos a realitzar. Aquest apartat se centra, principalment, en la feina que es volia fer des d'un principi i que no s'ha pogut aconseguir per diversos factors, però també s'hi planteja una idea per seguir la recerca en aquest camp.

11.1 Tractar els rígids

Tal com s'ha comentat en l'apartat 9.1.5, els fragments rígids d'un arbre RPST són complexos de tractar, ja que en contenir cicles no hi ha una forma fàcil de descriure'ls en text natural. Per això, una de les possibles millores a fer seria dissenyar una estratègia per a tractar-los.

Artem Polyvyanyy desenvolupa en la seva tesi doctoral [55] un mètode que reestructura els models amb rígids i els converteix en versions equivalents ben estructurades.

Un procés està ben estructurat, si per a cada node amb múltiples arcs de sortida (bifurcació), hi ha un node corresponent amb múltiple arcs d'entrada (unió) i viceversa, de manera que el fragment del model entre la bifurcació i la unió forma un component *single-entry-single-exit* (SESE) [56]. Si no es compleix aquesta condició, es considera que el model és desestructurat.

Malgrat tot, no tots els models es poden convertir, per tant, prèviament cal comprovar que es puguin convertir.

La llibreria de *Java* que es pot fer servir es diu *bpstruct* [56]. Aquesta eina transforma els processos desestructurats i els estructura correctament.

És una millora que permetria augmentar la quantitat de diagrames BPMN que es poden processar correctament amb aquest programa. Això seria interessant perquè, actualment,

si es vol fer servir el conjunt de dades per a entrenar un model d'aprenentatge automàtic, serà molt complicat que el model sigui capaç de generar correctament els BPMN amb rígids.

11.2 Aconseguir resultats més naturals

Una altra millora consistiria a implementar estratègies perquè les descripcions resultants siguin menys automatitzades, més variades i semblin més reals. Cal tenir en compte que desenvolupar aquest canvi pot millorar la qualitat del model generat amb el *dataset* resultant.

Hi ha diverses formes d'aconseguir-ho, però una senzilla d'implementar seria ampliar la quantitat de connectors que es fan servir en l'apartat 9.3.4.

11.3 Crear model Deep Learning

Finalment, per a continuar investigant en el camp dels models BPMN, es proposa crear un model *Deep Learning* que rebi una descripció en llenguatge natural d'un BPMN i el transformi en el diagrama corresponent. Aquest model s'entrenaria amb les dades que s'han generat en aquest treball.

Inicialment, es va considerar fer una petita prova del conjunt de dades amb un model senzill, però per manca de temps no s'ha pogut dur a terme. Realitzar aquesta prova seria important per a validar el *dataset*, o bé, saber si cal fer-hi modificacions.

Aconseguir tenir un model fiable que converteixi de text a diagrama, seria molt positiu per a tota persona que treballi dissenyant diagrames BPMN. Li estalviaria temps i esforços en dibuixar el diagrama i, per tant, podria dedicar-se, plenament, a descriure el procés.

12

Conclusions

Un cop acabat el projecte, és important fer una anàlisi de com ha anat el treball i extreure'n conclusions i fer una autocrítica que ha de servir per millorar en treballs futurs. En aquest apartat s'avaluen qüestions, a escala personal, com la gestió del temps, els coneixements adquirits. També es fa un breu resum de les principals contribucions d'aquest treball.

12.1 Gestió del temps

Primer de tot, cal remarcar que la gestió del temps en aquest treball ha sigut deficient. En la planificació inicial 6 es va fallar a l'hora de calcular les hores disponibles per a desenvolupar el treball.

No es va tenir prou en compte que treballar a temps complet, estudiar una assignatura de la universitat i fer el treball eren incompatibles.

Mentre va existir la càrrega de l'assignatura no es va avançar pràcticament res amb el projecte. Més tard, el projecte es va poder dur a terme de forma continuada, però seguir amb la feina ha implicat no poder-hi dedicar moltes hores setmanalment. Tot això ha comportat diversos endarreriments en la data d'entrega. De cara al futur, aquesta gestió del temps cal millorar-la per a evitar aquests retards en nous projectes.

12.2 Metodologia

El fet de treballar en una empresa durant el treball també ha aportat coses bones. Inicialment, el coneixement de metodologies de treball era gairebé escàs. En la secció de metodologia 5 s'explica com s'escull *Kanban*. Aquesta és totalment vàlida per aquest projecte, però com que no es tenia prou coneixement sobre eines de gestió de projectes,

és possible que no s'hagi aprofitat tot el seu potencial.

En la feina es treballava amb *Scrum* (molt similar a la que fem servir, però més rígida) i això ha permès descobrir noves eines, com *Azure DevOps*, que estan especialitzades en *Agile*. Aquests instruments permeten controlar i categoritzar millor la feina a realitzar. I, per tant, saber fàcilment quines tasques cal prioritzar.

12.3 Coneixements adquirits

El projecte ha servit per millorar molt a títol personal. Durant el desenvolupament s'han après nocions de gestió de projectes, com decidir què s'ha d'implementar i què no, prioritzar la feina, comunicar-se amb els directors del projecte i corregir la feina a partir del seu *feedback*.

També s'ha millorat la capacitat d'organitzar un projecte de *software* i també 3 de programar en *Java*. Al principi, tractar amb aquest llenguatge va ser complicat perquè es tenia un nivell bàsic. Però, a poc a poc s'ha après a fer-lo anar millor. A més a més, el descobriment i aprenentatge de les llibreries *Camunda*, *Freeling*, *SimpleNLG* i *jBPT* ha sigut molt positiu. Ja que, és possible que algun dia es dugui a terme algun projecte personal amb alguna d'aquestes eines.

I finalment, s'han adquirit molts coneixements sobre els models BPMN. Quan es va començar el treball no es coneixia l'existència d'aquests. Gràcies al treball s'ha fet una recerca que ha portat a aprendre què són, per a què es fan servir i com fer-los servir en projectes software.

12.4 Aportacions d'aquest treball

En aquest treball de fi de grau s'ha aconseguit crear un programa que, donada una entrada de diversos models BPMN, generi una descripció en llenguatge natural per a cada un dels models.

El programa llegeix (*parseja*) els models, analitza morfosintàcticament les frases dels elements, genera unes frases simples fent servir l'anàlisi morfosintàctica i finalment, genera la descripció unint les frases i afegint connectors entre elles perquè quedi més natural.

També s'ha creat un *dataset*, format per parells de models BPMN i les seves descripcions en llenguatge natural. Aquest conjunt de dades es pot fer servir per a entrenar un model de *Deep Learning*.

Per a desenvolupar el programa s'han estudiat i emprat diverses eines que han ajudat a obtenir un millor resultat. Per llegir els models s'ha fet servir *Camunda*, per fer l'anàlisi

morfosintàctica s'ha fet anar *Freeling*, per crear i combinar frases simples s'ha fet servir *SimpleNLG* i per a generar i recórrer arbres RPST s'ha utilitzat *jBPT*.

Després d'haver utilitzat aquestes eines, es pot concloure que la inclusió d'eines de NLP i NLG, com són *Freeling* i *SimpleNLG*, ha permès obtenir un resultat més real en la generació de descripcions en llenguatge natural a partir de models BPMN. Això demostra el potencial que tenen aquestes llibreries quan s'utilitzen juntes en la generació de qualsevol text en llenguatge natural.

El programa s'ha dissenyat de forma modular. De tal manera que cada secció realitza una funció independentment de les altres. Per tant, resulta senzill aprofitar i reutilitzar algun dels mòduls sense necessitat de fer-los servir tot.

A més a més, tal com s'explica en l'apèndix B, el codi està publicat de forma *open source*. Per tant, qualsevol que ho consideri pot consultar-lo o modificar-lo.

Bibliografia

[1] Institut d'Estudis Catalans. Diec2. https://dlc.iec.cat/. Accedit el 26/09/2021.

[2] Ashley DiFranza. What does a business analyst do? https://www.northeastern.edu/graduate/blog/what-does-a-business-analyst-do/. Accedit el 26/09/2021.

[3] Wikipedia. Business process model and notation. https://en.wikipedia.org/wiki/Business_Process_Model_and_Notation. Accedit el 20/08/2021.

[4] Beyond BPMN. Beyond bpmn. https://beyondbpmn.com/. Accedit el 24/08/2021.

[5] Marin Perez. The beginner's guide to using bpmn in business. https://www.microsoft.com/en-us/microsoft-365/business-insights-ideas/resources/the-guide-to-using-bpmn-in-your-business. Accedit el 24/08/2021.

[6] Visual Paradigm. Bpmn activity types explained. https://www.visual-paradigm.com/guide/bpmn/bpmn-activity-types-explained/. Accedit el 16/10/2021.

[7] Natural language processing. https://en.wikipedia.org/wiki/Natural_language_processing. Accedit el 01/06/2022.

[8] What is linguistics? — linguistic society of america. https://www.linguisticsociety.org/what-linguistics.

[9] What is natural language processing? intro to nlp in machine learning. https://www.gyansetu.in/what-is-natural-language-processing/. Accedit el 01/06/2022.

[10] Gonz Saavedra. Natural language processing (nlp): What is it & how does it work? https://monkeylearn.com/natural-language-processing/. Accedit el 01/06/2022.

[11] Parsing. https://en.wikipedia.org/wiki/Parsing. Accedit el 01/06/2022.

[12] Jeremy Gallemard. What is semantic analysis? https://blog.smart-tribune.com/en/semantic-analysis, 2022. Accedit el 01/06/2022.

[13] Gonz Saavedra. Nlp, machine learning & ai, explained. https://monkeylearn.com/blog/nlp-ai/. Accedit el 01/06/2022.

[14] What is natural language processing? https://www.ibm.com/cloud/learn/natural-language-processing. Accedit el 01/06/2022.

[15] nltk. Nltk source. https://github.com/nltk/nltk. Accedit el 01/06/2022.

[16] Natural language generation. https://en.wikipedia.org/wiki/Natural_language_generation. Accedit el 08/06/2022.

[17] Josep Curto. ¿qué es nlg? - tecnología++. https://blogs.uoc.edu/informatica/que-es-nlg/, 2017. Accedit el 06/06/2022.

[18] Eda Kavlakoglu. Nlp vs. nlu vs. nlg: the differences between three natural language processing concepts - watson blog. https://www.ibm.com/blogs/watson/2020/11/nlp-vs-nlu-vs-nlg-the-differences-between-three-natural-language-processing-concepts 2020. Accedit el 06/06/2022.

[19] Sciforce. A comprehensive guide to natural language generation. https://medium.com/sciforce/a-comprehensive-guide-to-natural-language-generation-dd63a4b6e548, 2018. Accedit el 06/06/2022.

[20] Cem Dilmegani. Natural language generation (nlg): What it is & how it works. https://research.aimultiple.com/nlg/, 2019. Accedit el 08/06/2022.

[21] Arria. Arria nlg for business intelligence. https://www.arria.com/. Accedit el 08/06/2022.

[22] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. 2020.

[23] Ehud Reiter. simplenlg. https://github.com/simplenlg/simplenlg.

[24] Matthew Stewart. The limitations of machine learning. https://towardsdatascience.com/the-limitations-of-machine-learning-a00e0c3040c6. Accedit el 15/03/2022.

[25] Genís Martín Coca. Transformant models bpmn a llenguatge natural, 2016.

[26] Josep Carmona, Lluís Padró, Marc Solé, and Sànchez-Ferreres Josep. Process talks. https://www.processtalks.com/. Accedit el 26/09/2021.

[27] Fabian Friedrich, Jan Mendling, and Frank Puhlmann. Process model generation from natural language text. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6741 LNCS:482–496, 2011.

[28] Lluís Padró, Miquel Collado, Samuel Reese, Marina Lloberes, and Irene Castellón. Freeling 2.1: Five years of open-source language processing tools. In *Proceedings of 7th Language Resources and Evaluation Conference (LREC'10)*, La Valletta, Malta, May 2010.

[29] Lluís Padró. Analizadores multilingües en freeling. *Linguamatica*, 3(2):13–20, December 2011.

[30] Lluís Padró and Evgeny Stanilovsky. Freeling 3.0: Towards wider multilinguality. In *Proceedings of the Language Resources and Evaluation Conference (LREC 2012)*, Istanbul, Turkey, May 2012. ELRA.

[31] Xavier Carreras, Isaac Chao, Lluís Padró, and Muntsa Padró. Freeling: An open-source suite of language analyzers. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04)*, 2004.

[32] Jordi Atserias, Bernardino Casas, Elisabet Comelles, Meritxell González, Lluís Padró, and Muntsa Padró. Freeling 1.3: Syntactic and semantic services in an open-source nlp library. In *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy, May 2006. ELRA.

[33] Max Rehkopf. What is a kanban board? — atlassian. `https://www.atlassian.com/agile/kanban/boards`. Accedit el 18/12/2021.

[34] Wikipedia. Agile software development. `https://en.wikipedia.org/wiki/Agile_software_development`. Accedit el 28/09/2021.

[35] Wikipedia. Version control. `https://en.wikipedia.org/wiki/Version_control`. Accedit el 16/10/2021.

[36] Wikipedia. Git. `https://en.wikipedia.org/wiki/Git`. Accedit el 16/10/2021.

[37] Wikipedia. Github. `https://en.wikipedia.org/wiki/GitHub`. Accedit el 16/10/2021.

[38] Todoist. Features. `https://todoist.com/features`. Accedit el 16/10/2021.

[39] Wikipedia. Gantt chart. `https://en.wikipedia.org/wiki/Gantt_chart`. Accedit el 03/10/2021.

[40] Glassdoor. Sueldo: Programador en barcelona. `https://www.glassdoor.es/Sueldos/barcelona-programador-sueldo-SRCH_IL.0,9_IM1015_KO10,21.htm?clickSource=searchBtn`. Accedit el 10/10/2021.

[41] CSIF, APT, UGT, CCOO, and Intersindical-CSC. Comunicado conjunto: Negociación de teletrabajo. https://csifcapgemini.blogspot.com/2021/10/comunicado-conjunto-negociacion-de.html. Accedit el 10/10/2021.

[42] Instituto universitario de investigación en Ciencia y Tecnologías de la Sostenibilidad. Edinsost, proyecto de innovación educativa — instituto universitario de investigación en ciencia y tecnologías de la sostenibilidad — upc. universitat politècnica de catalunya. https://is.upc.edu/es/noticias/edinsost. Accedit el 15/10/2021.

[43] Software Design Principles. Top 5 principles of software development. https://www.educba.com/software-design-principles/. Accedit el 09/03/2022.

[44] Hammad Madhuri. Principles of software design — geeksforgeeks. https://www.geeksforgeeks.org/principles-of-software-design/. Accedit el 09/03/2022.

[45] Camunda. Workflow and decision automation platform. https://camunda.com/. Accedit el 16/10/2021.

[46] Wikipedia. Camunda. https://en.wikipedia.org/wiki/Camunda. Accedit el 09/03/2022.

[47] Charles Humble. Camunda forks alfresco activiti. https://www.infoq.com/news/2013/03/Camunda-Forks-Activiti/. Accedit el 09/03/2022.

[48] TALP. Language and speech technologies and applications. https://www.talp.upc.edu/. Accedit el 10/03/2022.

[49] Brad Jascob. pysimplenlg. https://github.com/bjascob/pySimpleNLG. Accedit el 11/03/2022.

[50] Artem Polyvyanyy, Matthias Weidlich, Luciano García Bañuelos, Christian Wiggert, Martin Mader, and Gero Decker. jbpt/codebase. https://github.com/jbpt/codebase. Accedit el 22/03/2022.

[51] Jussi Vanhatalo, Hagen Völzer, and Jana Koehler. The refined process structure tree. *Data & Knowledge Engineering*, 68:793–818, 9 2009.

[52] Nikola Tanković. python-bpmn-engine. https://github.com/ntankovic/python-bpmn-engine. Accedit el 23/03/2022.

[53] bpmn miwg. bpmn-miwg-test-suite. https://github.com/bpmn-miwg/bpmn-miwg-test-suite, 2015. Accedit el 02/06/2022.

[54] Henrik Leopold, Heiner Stuckenschmidt, Matthias Weidlich, Christian Meilicke, and Elena Kuss. Process model matching contest @emisa. https://ai.wu.ac.at/emisa2015/contest.php, 2015. Accedit el 02/06/2022.

[55] Artem Polyvyanyy. *Structuring process models*. PhD thesis, University of Potsdam, 03 2012.

[56] Artem Polyvyanyy. bpstruct. https://code.google.com/archive/p/bpstruct/. Accedit el 31/05/2022.

Apèndix A

Models BPMN i descripcions

Apèndix B

Codi

Tot el codi desenvolupat en aquest projecte, els models utilitzats i les descripcions generades es poden trobar al següent repositori de Github `https://github.com/guillempla/TFG`. El codi està publicat sota la llicència GNU General Public License v3.0.