Proyecto de Grado

Edición 2019

- Citaciones en las sentencias de la Base de Jurisprudencia Nacional -

Andrés Fulloni[1]

Damián Langone[2]

24 de abril de 2020

Docente Supervisor:
Dina Wonsever

Montevideo, Uruguay
Instituto de Computación
Facultad de Ingeniería
UDELAR

[1] alfredo.andres.fulloni.papaleo@fing.edu.uy
[2] damian.langone@fing.edu.uy

Resumen

El proyecto presentado en este documento trata sobre la construcción de un sistema para el procesamiento de sentencias judiciales de la Base de Jurisprudencia Nacional (BJN), las cuales corresponden a sentencias de segunda instancia, es decir, apelaciones de sentencias anteriores. El principal factor de motivación para este trabajo fue poder generar herramientas para el mayor aprovechamiento de los datos. El proceso se dividió en tres etapas: extracción de referencias a distintos objetos legales presentes en los textos de las Sentencias, post-procesamiento y carga de la información extraída a una base de datos orientada a grafos y por último el etiquetado semántico de las referencias encontradas. Al realizar la primera parte se generaron recursos muy útiles para poder realizar otros estudios sobre distintos aspectos del funcionamiento del área legal. El resultado de la segunda etapa fue un grafo de referencias, el cual ofrece la capacidad de mejorar la interpretación de la información almacenada en la BJN, además de facilitar la manipulación de los datos y ofrecer la posibilidad de realizar análisis más profundos sobre la información de las sentencias.

Para la primera etapa se comenzó realizando un etiquetado manual de las distintas referencias presentes en los textos de la BJN, para luego abordar la tarea de extracción de referencias mediante distintos enfoques. El primer paso tomado consistió en la creación de reglas manuales mediante expresiones regulares para la extracción del texto completo de una referencia. Además se utilizaron otras herramientas como NeuroNER, Campos Aleatorios Condicionales (CRF) y SpaCy. Para esta etapa, mediante la utilización de esta última herramienta se obtuvo un 87.63 % de medida micro-F1. Además, dado que uno de los objetos de estudio son los Doctrinos del área legal, y debido a que estos se corresponden a personas, se utilizaron otros métodos para la extracción de referencias a los mismos con el objetivo de generar una lista primaria de nombres sobre los cuales se pueda trabajar en un futuro. Las herramientas utilizadas para esta tarea fueron el sistema "Identificación de opiniones de diferentes fuentes en textos en Español" y SPIED, un sistema de extracción de información desarrollado por el equipo de PLN de Stanford.

Para la segunda etapa se trabajó con el motor de bases de datos Neo4j, el cual brinda soporte para esquemas de información orientados a grafos. Para poder crear el grafo de referencias fue necesario realizar un post-procesamiento de la información extraída en la primera etapa. Para esto se desarrolló un módulo que toma como entrada al texto de una referencia y extrae mediante diferentes reglas las propiedades más importantes de la misma. Luego, con estas propiedades se construye dinámicamente una consulta que al ejecutarse crea los nodos y aristas necesarios. Al evaluar este módulo individualmente se obtuvo un 94.79 % de medida micro-F1. También se desarrolló una estrategia de evaluación del sistema completo, teniendo en cuenta los componentes desarrollados, con la cual se obtuvo un 89.05 % de medida micro-F1.

Finalmente, la tercera etapa consiste en un estudio sobre el etiquetado semántico de las referencias. Esto significa poder, de manera automática, asignarle un sentido semántico a cada referencia. Se estudiaron los aspectos correspondientes al posible desarrollo de esta tarea, preparando el camino para futuros trabajos en esta área. En esta etapa se pudieron reconocer diferentes problemas relacionados a la identificación de la razón por la cual se realiza una referencia, y a la identificación de predicados. Estos problemas requieren de más tiempo y conocimiento técnico del área legal para su abordaje. Se comentan brevemente los mismos y posibles pasos para completar la tarea.

Índice general

Resumen **2**

1. Introducción **11**

2. Marco de Trabajo **16**

2.1. Conceptos necesarios 16

2.2. Estado del Arte 18

3. Base de Jurisprudencia Nacional **24**

4. Expresiones Regulares y generación del Corpus **27**

4.1. Expresiones Regulares 28

4.1.1. Proceso de creación 28

4.1.2. Proceso de anotación 30

4.1.3. Evaluación y Resultados 31

4.2. Generación del sub-corpus 34

5. Extracción de referencias utilizando Aprendizaje Automático **35**

5.1. NeuroNER 36

5.1.1. Arquitectura 36

5.1.2. Características principales 38

5.1.3. Utilización 39

5.1.4. Separación de oraciones 39

5.1.5. Evaluación y Resultados 41

5.2. Campos Aleatorios Condicionales (CRF) 46

5.2.1. Utilización 46

5.2.2. Atributos seleccionados 46

5.2.3. Evaluación y Resultados 47

5.3. SpaCy 50
5.3.1. Utilización 50
5.3.2. Evaluación y Resultados 51

6. Otros procesos sobre la BJN **53**
6.1. Extracción de Doctrinos 53
6.1.1. Identificación de opiniones de diferentes fuentes en textos en Español 53
6.1.2. Obtención de candidatos a Doctrinos 55
6.1.3. Post Procesamiento de fuentes 56
6.1.4. SPIED: Stanford Pattern-based Information Extraction and Diagnostics 59
6.2. Análisis Semántico 61
6.2.1. Extracción de predicados 61
6.2.2. Etiquetado Semántico 64

7. Grafo de referencias **65**
7.1. Características 65
7.2. Herramienta 66
7.3. Proceso de carga de datos 70
7.3.1. Extracción de propiedades 71
7.3.2. Transformación de la información 77
7.3.3. Carga de la información 79
7.3.4. Ejemplo del proceso 81
7.4. Problemas 84
7.5. Resultados 87
7.6. Evaluación 90
7.6.1. Evaluación del Sistema de carga al Grafo 91
7.6.2. Evaluación del Sistema completo 95

8. Conclusiones y trabajo futuro **100**
8.1. Conclusiones 100
8.2. Trabajo futuro 101

Bibliografía **104**

Anexo A. Esquema de anotación BRAT **107**

Anexo B. Expresiones Regulares **109**

Anexo C. Extracción de Referencias **113**

Anexo D. Grafo de Referencias **115**

Anexo E. Lista de Cuerpos Normativos **121**

Índice de figuras

2.1. Comparación de crecimiento en popularidad. *Extraído de www.db-engines.com [1]* 20
2.2. Comparación bases de datos orientadas a grafos. Extraído de [2] 22

3.1. Información adicional de una Sentencia 24
3.2. Filtros en búsqueda selectiva 26

4.1. Ejemplo de aplicación de reglas 28
4.2. Ejemplo de coincidencias 29
4.3. Diagrama del proceso de extracción 30
4.4. Ejemplo de esquema de anotación para Artículos 30
4.5. Ejemplo salida BRAT 31

5.1. Arquitectura NeuroNER. *Extraído de www.neuroner.com [3]* 37
5.2. Fragmento de texto utilizado en la evaluación 40
5.3. Cantidad de referencias por conjunto 41
5.4. Ejemplos formatos Ley y Ficha 44
5.5. Ejemplos formatos Sentencia 45
5.6. Transiciones CRF 48
5.7. Atributos principales consideradas para Ley 48
5.8. Formato de entrada SpaCy 51
5.9. Medida F en función de epochs en validación 51

6.1. Ejemplos referencias a Doctrinos 54
6.2. Ejemplos de reglas simplificadas para fuente. Extraído de [4] 55
6.3. Ejemplos de fuentes reconocidas 56
6.4. Criterio para determinar Doctrino 58
6.5. Ejemplos referencias a Doctrinos 58

6.6. Ejemplos predicados 63

7.1. Resultado consulta Cypher en Neo4j 69
7.2. Flujo del sistema 70
7.3. Ejemplos referencias a Fichas 71
7.4. Ejemplos referencias a Sentencias 73
7.5. Asignación de año en múltiples Sentencias 74
7.6. Impacto de condiciones sobre tokens de una referencia 77
7.7. Creación nodo Ficha referenciado por Sentencia 79
7.8. Creación referencia Sentencia a Ficha 80
7.9. Creación nodo Artículo referenciado por Sentencia 80
7.10. Creación nodo Sentencia referenciado por otra Sentencia 81
7.11. Subgrafo ejemplo 83
7.12. Múltiples referencias 84
7.13. Plan de ejecución sin índices vs con índices 85
7.14. Estructura de Grafo 87
7.15. Consulta artículo más referenciado 88
7.16. Consulta artículo más referenciado del Código Penal 89
7.17. Consulta referencias a Doctrinos por Sede 89
7.18. Cantidad de referencias por Doctrino 89
7.19. Clasificación como descartado incorrectamente y descartado correctamente 90
7.20. Ejemplos descartados incorrectamente Artículo 92
7.21. Ejemplos incorrectos Artículo 93
7.22. Ejemplos incorrectos Sentencia 94
7.23. Ejemplos de evaluación completa 98

Índice de cuadros

4.1. Resultados expresiones regulares 32
4.2. Incompletas como FN 33
4.3. Incompletas como VP 33
4.4. Cantidad de referencias por conjunto 34

5.1. Ejemplo formas de evaluación 39
5.2. Superposición de referencias 42
5.3. Resultados SpaCy 43
5.4. Resultados Freeling 43
5.5. Resultados entrenando modelos separados 44
5.6. Resultados Sentencia evaluación BIO 45
5.7. Atributos del token 47
5.8. Atributos del token anterior y siguiente 47
5.9. Resultados CRF 48
5.10. Resultados SpaCy 52

6.1. Post-procesamiento de fuentes 57

7.1. Cantidad de nodos y relaciones salientes 88
7.2. Cantidad de referencias en evaluación individual 91
7.3. Resultados evaluación individual 95
7.4. Cantidades de referencias en evaluación completa 98
7.5. Resultados evaluación completa 99

Capítulo 1

Introducción

En el último tiempo la velocidad con la cual la tecnología se ha mezclado en diferentes áreas ha incrementado notoriamente. Actualmente se puede ver el uso de la misma en sectores en los cuales quizás algunos años atrás era impensado por los actores del mismo. Un ejemplo de sector en el cual la tecnología toma cada vez más protagonismo es el sector legal. Esto se debe a las múltiples herramientas con las cuales se cuenta para procesar grandes cantidades de datos y optimizar el aprovechamiento de los mismos.

En este trabajo se busca aplicar las herramientas disponibles para el Procesamiento de Lenguaje Natural, con el objetivo de poder aportar soluciones y avances para el manejo de la información de sentencias judiciales, más precisamente en sentencias judiciales almacenadas en la Base de Jurisprudencia Nacional (BJN). Un factor determinante presente en las sentencias judiciales, y sobre el cual se hace énfasis en este trabajo de investigación, es el correspondiente a las referencias entre objetos legales.

Motivación

Actualmente la información de la BJN se encuentra almacenada respetando un esquema relacional, lo cual lleva a que la unidad de información sea la sentencia en sí, impidiendo relacionar aspectos más específicos presentes en las redacciones de las mismas, como ser Artículos, Decretos o Leyes.

I) Sentencia 114/2017 del Tribunal de Apelaciones en lo Civil de 1er Turno.

   "Adhiere a la posición clásica que sostiene que en las hipótesis previstas por los ***arts. 24 y 25 de la Constitución****, los funcionarios públicos no están legitimados para ser demandados por los terceros perjudicados, pudiendo solo el Estado repetir contra los mismos en caso de ser responsabilizado..."*.

II) Sentencia 175/2009 del Tribunal de Apelaciones en lo Civil de 7mo Turno.

   "Por tanto, habiéndose promovido la demanda casi un año después del archivo penal y en conocimiento de la IMM desde octubre de 2006, no puede dudarse de la configuración de falta de servicio por su parte al no haberse pronunciado sobre el sumario ni la caducidad, tardando prácticamente un año más, lesionando los derechos del funcionario, por lo cual debe ser responsabilizada conforme a las disposiciones de los ***art. 24, 25 de la Constitución*** *de los daños y perjuicios derivados..."*.

Los ejemplos I) y II) se corresponden con extractos de texto de las sentencias 114 del año 2017 y 175 del año 2009. Estos segmentos de texto integran la redacción total de cada sentencia almacenada actualmente en la BJN. Como se puede ver, estas sentencias comparten referencias a los artículos 24 y 25 de la Constitución de la República, por lo que de alguna forma las mismas están relacionadas. Sin embargo, almacenar las redacciones de las mismas sin realizar ningún procesamiento, impide que esta referencia compartida se pueda identificar. Estas relaciones son a las que se apunta en este trabajo. Para poder aprovechar las mismas, es necesario poder reconocerlas, entenderlas y almacenarlas de una manera que permita el fácil flujo entre las mismas. Para esto se busca generar un sistema de extracción y categorización de referencias y aprovechar la utilización de grafos para el almacenamiento de la información, los cuales son conocidos por su utilidad en escenarios donde la información se encuentra muy relacionada.

Diferentes razones son las que llevan a pensar que un sistema de extracción y categorización de referencias en textos legales, y su almacenamiento en un esquema orientado a grafos puede llegar a ser de utilidad. Si bien existen sistemas de extracción de referencias a objetos legales en Español [5], no existe un sistema completo que trabaje sobre textos del área legal enfocado al sistema judicial Uruguayo. Con sistema completo, se hace referencia a aquél sistema capaz de, a partir de textos planos del área legal, extraer referencias, procesarlas y cargar la información en una base de datos que permita su interpretación para usuarios no especializados en el área. Debido a esto es que resulta interesante poder colaborar con esta rama de trabajo y poder allanar el camino para futuros proyectos.

También resulta de interés poder contar con las referencias identificadas para poder relacionarlas con la información de la fuente del objeto legal referenciado, lo cual mejoraría la interpretación de las Sentencias. Hoy en día, la BJN cuenta con la información almacenada pero sin sacarle mucho provecho, por lo que termina perdiendo valor. En cambio, las herramientas actuales presentes en el área de PLN permiten un aprovechamiento mucho más grande de la misma, por ejemplo, contar con las referencias cargadas en una base de datos orientada a grafos, permite realizar diferentes análisis sobre los objetos legales en sí, sobre el desarrollo del trabajo de distintos Tribunales o también sobre el desenlace de distintas sentencias judiciales. Además, un esquema orientado a grafos se adapta mejor al tipo de información contenido en estos documentos legales, permitiendo desarrollar a futuro sistemas donde cualquier usuario pueda ser capaz de realizar búsquedas e interpretar la información y las relaciones legales existentes.

Además de tener identificadas las propiedades de las referencias, también resulta de interés estudiar el significado semántico de las mismas. Es decir, la razón por la cual un determinado objeto legal es referenciado dentro de la redacción de una sentencia. Como se ve en el ejemplo II), la referencia está precedida por el texto “conforme a las disposiciones de los”, lo cual lleva a pensar que se está utilizando a dichos artículos como base legal para sostener un argumento u opinión acerca de determinado tema. Por lo tanto, reconocer los significados de las referencias resulta de utilidad para poder generar una clasificación más específica de los distintos tipos existentes y las situaciones en las cuales éstas aparecen. Esto permitiría, en el ámbito legal, facilitar el análisis de la utilización de los recursos legales existentes. Además, contar con una herramienta que solucione esta tarea, serviría como base para futuros proyectos que integren técnicas de PLN con problemas del área legal.

Otro factor de motivación al desarrollo de este proyecto es aportar a la mejora del motor de búsqueda actual de la BJN, en la cual la unidad de información es la sentencia, por lo que se accede a la información de forma individual. En este proyecto, las unidades de estudio son las referencias contenidas en dichas sentencias, las cuales son procesadas y almacenadas en un sólo grafo, permitiendo ver la conectividad entre las diferentes Sentencias.

Planteo del Problema

La extracción de información es una tarea que en los últimos años ha tenido un gran avance debido a su gran utilidad y a la gran cantidad de información que hoy en día es almacenada en diferentes ámbitos. Los problemas a tratar planteados en este proyecto se corresponden con diferentes sub áreas del Procesamiento de Lenguaje Natural (PLN).

El problema de extracción de referencias fue abordado como un problema de Reconocimiento de Entidades Nombradas (NER). Para esto, existen distintos enfoques que van desde sistemas basados en reglas hasta la utilización de modelos basados en Redes Neuronales. En el caso de la extracción de referencias de la BJN, la principal dificultad del problema pasa por la heterogeneidad de formatos en los cuales se pueden encontrar las mismas. Si bien este es un problema asiduo en las tareas de PLN, en este caso se ve acentuado debido a diferentes razones como la gran cantidad de Sentencias en la BJN (aproximadamente 72000), el gran período de tiempo entre la primera y la última (aproximadamente 30 años), la gran cantidad de diferentes redactores y el no seguimiento de pautas generales para la escritura de estos documentos.

Otro problema es el reconocimiento de nombres de Doctrinos y su distinción de los nombres de testigos u otros participantes del proceso judicial. Por esta razón, el problema no se puede abordar con un enfoque de NER, si no que se requiere la utilización de otras técnicas. En este proyecto, se tuvieron en cuenta diferentes factores como el sector dentro del texto de la Sentencia donde ocurre el nombre de la persona y los POS-Tag de las palabras vecinas a dicho nombre.

El segundo problema abordado en este proyecto se corresponde con la extracción de diferentes propiedades de las referencias. Su solución es puramente algorítmica, ya que se cuenta con el tipo de objeto legal referenciado (Artículo, Sentencia, Ley, Decreto, Ficha) y se conoce qué propiedades se requiere de los mismos. Por lo tanto la solución pasa por tener en cuenta las características de cada objeto y aplicar diferentes reglas para el reconocimiento de las propiedades requeridas.

Por último, la tercer etapa, correspondiente al análisis semántico de las referencias, cuenta con dos problemas. El primero consiste en la extracción de los predicados (o contexto) de las referencias, el cual se corresponde con un problema de etiquetado de secuencias de texto (Sequence labeling), mientras que el problema de etiquetado semántico se corresponde con un problema de clasificación multiclase. Para esto último, es necesaria la definición de un conjunto de etiquetas que puedan abarcar las diferentes razones de referenciado existentes en los textos de la BJN, donde una de ellas podría ser “base legal”, la cual se corresponde con el ejemplo presentado en la sección anterior.

Objetivos

Los objetivos planteados al comienzo de este proyecto fueron:

- Creación de un sistema de extracción de referencias a diferentes objetos legales en Sentencias de la BJN de forma automática.
- Creación de un grafo de referencias e investigación acerca de los beneficios de este tipo de almacenamiento de información.
- Definición de un conjunto de etiquetas semánticas adecuadas para el contexto de Sentencias judiciales. Creación de un sistema de etiquetado semántico de referencias de forma automática.

Resultados esperados

Al final de este proyecto se desea poder cumplir con los objetivos planteados y además poder contribuir con herramientas de utilidad para trabajos futuros sobre el área legal Uruguaya. Dentro de los resultados esperados, se encuentra la investigación del estado del arte en tareas como Reconocimiento de Entidades Nombradas, clasificación semántica, y tecnologías de almacenamiento orientadas a grafos.

Como parte de las herramientas construidas en el proceso de este trabajo se espera obtener un corpus anotado sobre sentencias de la BJN, conteniendo anotaciones de referencias a diferentes objetos legales. También se espera obtener una herramienta de extracción de referencias de forma automática, así como un grafo de referencias que contenga toda la información contenida en las sentencias de la BJN. A su vez, es de interés poder contar con un conjunto de etiquetas semánticas que abarquen las posibles razones por las cuales una referencia puede existir, y utilizarlo para la creación de un clasificador semántico de referencias.

Organización del Documento

Este documento está estructurado de manera de abarcar cada uno de los tres problemas a tratar en secciones individuales. Los capítulos fueron alineados de forma de comenzar con los primeros procesamientos de las Sentencias y culminar con la creación del grafo de referencias. También se cuenta con capítulos destinados al Estado del Arte, conceptos generales, conclusiones y trabajo a futuro, y un anexo.

Capítulo 1 Introducción acerca de los problemas a tratar en este proyecto, las motivaciones y los objetivos planteados en el mismo.

Capítulo 2 Se documenta la investigación acerca del Estado del Arte en las tareas de interés asociadas a los problemas a resolver. También se detallan diferentes conceptos necesarios para poder comprender la totalidad del proyecto.

Capítulo 3 Breve introducción sobre la Base de Jurisprudencia Nacional y los datos disponibles para trabajar.

Capítulo 4 Se documenta el proceso de creación de las reglas utilizadas para la generación del sub-corpus empleado en todo el proyecto.

Capítulo 5 Se detallan los diferentes enfoques tomados para la extracción de referencias de sentencias de la BJN de forma automática, plasmando los resultados obtenidos con cada uno.

Capítulo 6 Capítulo donde se detallan otros procesos realizados sobre la BJN, como ser la extracción de referencias a Doctrinos y distintos enfoques para la clasificación semántica de referencias, indicando los problemas existentes en el contexto de este trabajo.

Capítulo 7 Capítulo destinado a documentar la etapa de creación del grafo de referencias, indicando los detalles del procesamiento de la información, los resultados obtenidos y su utilidad.

Capítulo 8 Capítulo que contiene las conclusiones más relevantes obtenidas luego de culminar la realización de este proyecto, así como diferentes puntos a tratar en el futuro sobre la línea de este trabajo.

Bibliografía Se presentan las referencias bibliográficas utilizadas en este proyecto.

Anexo Sección en la cual se adjuntan ejemplos sobre distintas problemáticas tratadas, así como información adicional sobre algunos puntos trabajados.

Capítulo 2

Marco de Trabajo

2.1. Conceptos necesarios

En esta sección se hace una introducción a los diferentes conceptos manejados en este proyecto. Algunos de ellos están relacionados al área legal, y más precisamente a los elementos utilizados en este trabajo. La unidad de estudio en este proyecto son las Sentencias de la Base de Jurisprudencia Nacional (BJN) [6] y los diferentes elementos que ocurren en sus redacciones.

BJN La Base de Jurisprudencia Nacional es un sistema público que permite el almacenamiento y la búsqueda de fallos judiciales de Segunda Instancia.

Objeto legal En el marco de este trabajo el término "objeto legal" se utiliza para hablar sobre los elementos del área legal que pueden ser referidos dentro de una Sentencia.

Sentencia Una sentencia es una resolución de carácter jurídico que expresa una decisión definitiva sobre un proceso. Existen varias categorizaciones de Sentencias, dentro de las cuales se encuentran las de Primera Instancia y las de Segunda Instancia. Estas últimas se corresponden a apelaciones a Sentencias de Primera Instancia y son el elemento principal de este trabajo.

Artículo Son las disposiciones, generalmente enumeradas de forma consecutiva, que conforman un cuerpo legal. Estos artículos pueden estar divididos en numerales, incisos y literales.

Ley Son reglas o normas establecidas por una autoridad superior para regular, de acuerdo con la justicia, algún aspecto de las relaciones sociales.

Decreto Decisión del Consejo de Ministros, o de una entidad equivalente, que aprueba disposiciones de carácter general.

Ficha o Expediente Reunión de documentos, escritos de procedimiento y fallos, relativos a un litigio incoado ante una jurisdicción civil, comercial o social, dentro de un legajo en el cual se mencionan los distintos acontecimientos del proceso.

Doctrino Se usa en el dominio legal para hacer referencia a juristas de gran prestigio, cuando se citan sus opiniones sobre asuntos con alguna vinculación con el que se trata.

Cuerpo Normativo Toda aquella fuente de derecho, como por ejemplo Código General del Proceso y Constitución de la República.

Sede Se denomina Sede al órgano judicial encargado de emitir las Sentencias. En el marco de este trabajo las sedes pueden ser Juzgados, Tribunales o mismo la Suprema Corte de Justicia.

Referencia En este trabajo se utiliza la palabra "referencia" cuando se hace mención a un objeto legal dentro del texto de una Sentencia. Un ejemplo de referencia al objeto legal Artículo se ve en "*..La Sala coincide con el apelante en que, de acuerdo con el art. 1608 del C. de Comercio, no corresponde condenar al pago de los intereses...*".

Precision Métrica utilizada en problemas de Extracción de Información para medir qué proporción de los ejemplos extraídos, son correctos. Es decir, de todo lo que se extrajo, qué proporción fue extraída correctamente.

Recall También llamada "Recuperación", es una métrica utilizada en problemas de Extracción de Información para medir qué proporción de los ejemplos totales, fueron extraídos correctamente. Es decir, de todo lo que se debería haber extraído, cuánto realmente se extrajo correctamente.

Medida F1 Es una métrica utilizada en problemas de Extracción de Información que se corresponde con la media armónica entre la Precision y el Recall. Es la medida utilizada para obtener una visión del comportamiento general de un modelo, ya que penaliza el desequilibrio entre las dos medidas mencionadas anteriormente.

2.2. Estado del Arte

Antes de comenzar con el desarrollo de este proyecto es necesario investigar acerca de las metodologías y herramientas más utilizadas para cada tarea. como se mencionó anteriormente, este proyecto se divide principalmente en tres problemas: la extracción de referencias, el manejo de sistemas de almacenamiento orientados a grafos y la clasificación semántica, tarea que incluye a la extracción de predicados.

Extracción de referencias

Al considerar a la tarea de extracción de referencias como un problema de NER, es necesario tener en cuenta las distintas opciones empleadas para este tipo de problemas. A grandes rasgos, se pueden identificar dos enfoques: sistemas basados en reglas o métodos estadísticos. El principal antecedente sobre el cual se basa este trabajo, utiliza un sistema basado en reglas manuales mediante expresiones regulares para la extracción de referencias (Sadeghian et al. [7]). Sin embargo, no aportan medidas de evaluación sobre las reglas creadas pero hacen referencia al método propuesto por Tran et al. [8]. En dicho trabajo se basan en la hipótesis de que las referencias en el dominio legal poseen su propia estructura y difieren de las referencias en otros dominios. Distinguen dos partes por las que se compone una referencia: la "posición" y el "contenido", siendo la primera interpretable por expresiones regulares. A su vez definen varios tipos de clasificación para estos segmentos de textos, el primero corresponde a las posiciones bien formadas, es decir que contienen un término identificable (como por ejemplo la palabra artículo), seguido de un número. Este tipo contiene toda la información necesaria para identificar a la referencia. Otros tipos tenidos en cuenta son anáforas y posiciones coordinadas, es decir cuando varios elementos son referidos conjuntamente. El estudio se realizó sobre La Ley Nacional de Pensiones de Japón (JNPL), sobre un total de 99 artículos y 748 citaciones y para la tarea de reconocimiento de referencias obtuvieron un 96.18 % de accuracy.

Otro trabajo que utiliza un sistema basado en reglas se presenta en Adedjouma et al. [9], en el cual los autores utilizan gramáticas con diversos patrones para la identificación y posterior extracción de las referencias. Este trabajo se realizó sobre textos legislativos de Luxemburgo. Para estructurar las gramáticas utilizadas, los autores diferencian entre referencias individuales y referencias compuestas (referencias que contienen más de una referencia individual). Luego, mediante diferentes patrones definidos en sus gramáticas, extraen las referencias compuestas de los textos con unas medidas de 99.7 % de precisión, 97.9 % de recall y 98.8 % de medida F1. A su vez, estas referencias compuestas son procesadas para resolver las referencias individuales contenidas en las mismas. Esto se realiza mediante patrones más específicos en las gramáticas, con los cuales se obtienen 99.9 % de precisión, 97.5 % de recall y 98.7 % de medida F1.

En cuanto a trabajos realizados sobre el idioma Español, se destaca el de Badji [5], donde se presenta un sistema basado en reglas para extracción de referencias legales tanto en Español como Inglés utilizando expresiones regulares el cual obtuvo un resultado de 85 % y 95 % de medida F1 respectivamente. Cabe destacar que para el Español el estudio se realizó sobre documentos formales como contratos y en informales como textos extraídos de Twitter. Un punto en común entre la documentación de los trabajos que utilizaron sistemas basados en reglas es la mención de la dificultad que generan las diferentes formas de citaciones existentes, lo cual provoca que la cobertura obtenida con estos sistemas sea menor.

Por otro lado, en el área de los métodos estadísticos se destacan distintas variantes de modelos de Campos Aleatorios Condicionales (CRF). En un trabajo con tareas similares al presente (Leitner, Elena et al. [10]) se tomaron como principales enfoques a los modelos CRF y a las redes neuronales BiLSTM (Bi-directional Long Short Term Memory). El estudio se realizó sobre un dataset de sentencias de la corte Alemana, el cual contiene alrededor de 67000 sentencias y 54000 entidades anotadas (además de entidades legales se tomaron en cuenta categorías más genéricas como ciudad, marca o institución) En total se probaron con 12 modelos y uno de los destacados fue una variación de CRF con 93.23 % de medida F1. En cuanto a los modelos BiLSTM, el que obtuvo mejor rendimiento fue un BiLSTM utilizado con Character Embeddings, el cual obtuvo 95.46 % de medida F1.

En el trabajo de Angelidis et al. [11], una de las categorías reconocidas y extraídas son las referencias legales. Este trabajo se basa en el idioma griego y utiliza Documentos legales publicados por el gobierno de Grecia. En dicho proyecto se hizo énfasis en los modelos BiLSTM probando tres variantes del mismo. Uno de ellos utiliza una capa BiLSTM junto con Word Embeddings y una capa de Regresión Logística con activación Softmax encargada de asignar las probabilidades de que el token pertenezca a cada clase. La segunda variante agrega otra capa BiLSTM mientras que la tercer variante utiliza una capa BiLSTM y reemplaza la capa de Regresión Logística por una capa CRF. El modelo que obtuvo mejores resultados en medida macro-F1 fue el que posee dos capas BiLSTM obteniendo un 88 %.

Cabe destacar que en cada citación realizada a trabajos anteriores se remarca el lenguaje sobre el cual trabajaron los autores debido a que, de acuerdo a Konkol [12], el idioma es uno de los factores más determinantes en esta tarea. En los sistemas basados en reglas, se necesitan realizar costosas modificaciones de las mismas para adaptarlas a otro idioma, mientras que en los otros métodos, esta tarea es más sencilla. Por ejemplo en métodos como CRF, donde se utilizan atributos manuales, se pueden seleccionar algunos que sean independientes del idioma, como ser las categorías gramaticales (POS-tags). El autor afirma que el rendimiento de los sistemas NER puede variar en más un 20 % dependiendo el idioma sobre el cual se trabaja. A su vez, coincidiendo con otros autores, remarca que un factor importante en el rendimiento de los sistemas NER son las entidades y el contexto sobre el cual se trabaja. Es por esta razón que al realizar la investigación del Estado del Arte se dirigió la búsqueda principalmente a trabajos relacionados con el área legal.

Bases de datos orientadas a grafos

La utilización de bases de datos orientadas a grafos ha ido incrementándose en el último tiempo como se puede ver en la figura 2.1, debido a su facilidad de aplicación en escenarios donde los datos se encuentran muy relacionados y estas relaciones presentan diversas propiedades que son de interés analizar. Esta tecnología está alterando muchas áreas, como lo son la gestión de la cadena de suministro, recomendaciones en comercio electrónico, seguridad, detección de fraude y muchas otras áreas en el análisis de datos avanzados.

Existen muchos trabajos que comparan las bases de datos relacionales y no relacionales con las orientadas a grafos. La principal comparación se da entre esquemas relacionales y esquemas orientados a grafos. La gran diferencia entre estos enfoques es la estructura en la cual se almacenan los datos y por ende el trabajo de

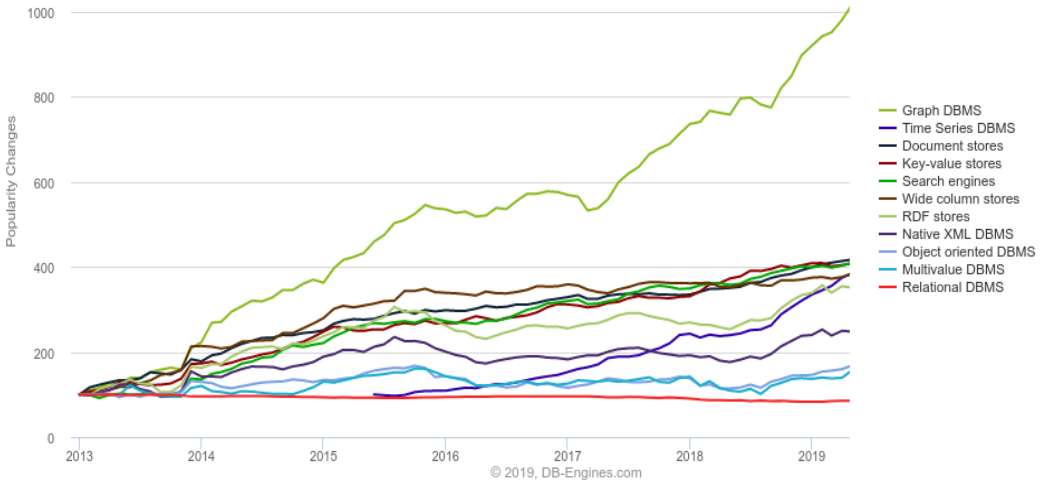

Figura 2.1: Comparación de crecimiento en popularidad. *Extraído de www.db-engines.com [1]*

diseño previo que se requiere para que el sistema de almacenamiento sea coherente y tenga el funcionamiento deseado. Este trabajo previo es muy costoso para un esquema relacional como se indica en [13], donde se plantea que una posible consulta sobre un sistema podría querer saber cuántas personas que compraron una tostadora, viven en Kansas, tienen antecedentes penales, y usaron cupones de descuentos para comprar el electrodoméstico. Si al momento de diseñar el esquema relacional, no se tuvo en cuenta un escenario similar al descripto, será muy difícil poder obtener la información mediante una consulta. En cambio en las bases de datos orientadas a grafos este tipo de consulta podría satisfacerse aunque no se haya tenido en cuenta desde un primer momento. Esto lleva al principal punto a favor de los esquemas orientados a grafos, la flexibilidad (Robinson et al. [14]). Este tipo de sistemas de almacenamiento permiten poder realizar adaptaciones a medida que el negocio cambia, permitiendo de esta forma agregar nuevos conceptos y relaciones sin que la funcionalidad del sistema se vea afectada. Esta flexibilidad también lleva a que no se presente el problema de los valores nulos como en el caso de las bases de datos relacionales, debido a que se brinda soporte para que nodos del mismo tipo puedan tener diferentes atributos. Otra característica importante relacionada con la flexibilidad está dada por el tamaño variable de los registros, lo cual evita problemas al ingresar datos de tamaños muy diferentes así como también ahorra el trabajo de definir el tamaño y tipo de los atributos.

Otro punto a favor remarcado en el libro citado se relaciona con el rendimiento. Las bases de datos de grafos permiten realizar consultas complejas de manera eficiente debido a que las relaciones entre los datos están definidas a nivel estructural. Esto es una ventaja con respecto a las bases de datos relacionales ya que estas deben realizar diferentes operaciones entre tablas para buscar las relaciones entre los datos, lo cual se realiza cada vez que se ejecuta una consulta. Esto no pasa con las bases de datos orientadas a grafos, donde el tiempo de ejecución para cada consulta es proporcional únicamente al tamaño del subgrafo recorrido para satisfacer la consulta, en lugar del tamaño del grafo completo.

También existen estudios que comparan los diferentes enfoques de estructuras de almacenamiento de datos desde una perspectiva más técnica. Un ejemplo de esto se ve en Vicknair et al. [15], donde se realiza

una comparación entre MySQL y el motor de base de datos orientado a grafos Neo4j. En dicho estudio se utilizaron diferentes medidas la velocidad de procesamiento sobre determinadas consultas, requerimientos de espacio en disco y escalabilidad. El experimento se realizó sobre varias bases de datos con distinta cantidad de información cargada. En cuanto al espacio requerido para almacenar la información, las bases de datos Neo4j llegaron en algunos casos hasta duplicar el tamaño ocupado por las bases de datos relacionales. Al momento de ejecutar diferentes consultas para medir la velocidad de los motores, se obtuvieron mejores resultados para Neo4j cuando se trataba de consultas estructurales, por ejemplo encontrar todos los nodos huérfanos. Por otro lado, cuando los datos eran numéricos, MySQL obtuvo mejores resultados ya que la búsqueda en Neo4j por defecto trata a los datos como texto, por lo que se deben realizar conversiones a la hora de comparar los valores, lo cual enlentece el proceso, mientras que cuando los datos eran textos, a medida que el tamaño de los mismos y la cantidad de información aumentaba, Neo4j aumentaba la diferencia a su favor en comparación con MySQL.

En [14], se marcan como puntos fundamentales a la hora de decidir entre diferentes tecnologías de bases de datos orientadas a grafos: la forma de almacenamiento interno y el motor de procesamiento. Según estos puntos, se pueden diferenciar dos tipos de bases de datos, las nativas y las no nativas. Las denominadas nativas se caracterizan por el uso de estructuras de almacenamiento optimizadas y diseñadas para la gestión de grafos, mientras que las no nativas presentan la idea de gestión de grafos pero los datos se terminan almacenando de la forma tradicional. El otro punto está relacionado a cómo se procesan las operaciones de la base de datos, donde la principal característica está dada por la adyacencia sin índice presente en las bases de datos nativas. Adyacencia sin índice significa que donde se almacena un nodo en memoria, también se almacena la dirección de memoria de cada uno de sus vecinos. Es decir, se almacenan en memoria los nodos junto a las relaciones con sus nodos vecinos. Las bases nativas, tienen un mejor rendimiento en el procesamiento debido a que garantizan que cada nodo estará almacenado junto a las referencias a sus nodos adyacentes. Esto permite que al momento de procesar una consulta, la adyacencia sin índice garantice un mejor tiempo de recuperación ya que no tiene una gran dependencia sobre índices, si no que se puede navegar entre nodos directamente debido a la forma en la que estos están almacenados en memoria. Por otro lado, las bases de datos no nativas generalmente utilizan una gran cantidad de índices lo que enlentece el procesamiento.

Siguiendo la línea de la comparación entre bases de datos orientadas a grafos, se destaca el trabajo de Fernandes et al. [2], donde se realiza una comparación entre las bases de datos AllegroGraph, ArangoDB, InfiniteGraph, Neo4J y OrientDB. En dicho estudio se tienen en cuenta factores como la escalabilidad, flexibilidad del esquema, lenguaje de consulta y la distribución de los datos. En la figura 2.2 se pueden ver los resultados obtenidos, en la escala de 0 (no soportado) a 4 (muy bueno), para diferentes propiedades de las bases de datos.

Los autores afirman que aunque la diferencia entre estas bases de datos no es muy grande y que la mayoría posee las propiedades muy bien implementadas, ArangoDB y a Neo4J se destacan por sobre las otras. De igual manera, recomiendan la utilización de Neo4J ya que presenta las funcionalidades mejor implementadas sobre ArangoDB. Además, de acuerdo a *www.db-engines.com* [1], Neo4J es la base de datos orientada a grafos más utilizada hoy en día, lo que reafirma la recomendación de los autores.

	AllegroGraph	ArangoDB	InfiniteGraph	Neo4J	OrientDB
Flexible Schema	1	3	3	4	3
Query Language	3	3	3	4	3
Sharding	3	3	0	0	3
Backups	3	2	3	4	3
Multimodel	4	4	2	2	4
Multi Architeture	3	4	3	4	3
Scalability	3	4	3	4	3
Cloud Ready	3	3	4	4	3
Total	23	26	21	26	25

Figura 2.2: Comparación bases de datos orientadas a grafos. Extraído de [2]

Análisis semántico

Para completar el alcance de este proyecto, faltaría abordar el problema del análisis del comportamiento semántico de las referencias extraídas. Es decir, identificar por qué se está realizando la citación. Para esto, se pueden tomar como base diversos trabajos dentro de los que se destacan el de Sadeghian et al. [7], el cual separa el problema en dos subtareas. La primera consiste en detectar los predicados de las citaciones y la segunda en clasificar cada una de ellas. En cuanto a la primera subtarea, los autores proponen utilizar un CRF con etiquetado BIO con diferentes atributos como ser categorías gramaticales o posición del token dentro de la sentencia. Luego, partiendo de un conjunto de predicados anotados manualmente entrenaron el modelo y llegaron, para la medida F1, a 87.5 % para la etiqueta B, 89.8 % para la etiqueta I y 99.8 % para la etiqueta O. No se encontraron otros estudios específicos que trataran la extracción de predicados de citaciones en textos legales, pero se observó que el problema de extracción de predicados era siempre abordado por las herramientas del estado del arte asociadas a los problemas de NER, siendo algunas de estas las mencionadas en la sección 2.2.

En cuanto a la tarea de clasificar semánticamente cada referencia, es un común denominador entre los trabajos investigados, la definición de un conjunto de etiquetas semánticas acorde al corpus sobre el cual se está trabajando. En el estudio al que se está haciendo referencia se proponen estrategias de aprendizaje supervisado como Support Vector Machine (SVN) y Regresión Logística (LR), y estrategias de aprendizaje no supervisado como k-Means, en las cuales se utilizan Word Embbedings como atributos de las palabras a clasificar. El modelo que mejor rendimiento obtuvo fue SVN con un valor de accuracy de 79 % evaluando, no sólo con la etiqueta más probable determinada por el modelo, si no que con las dos más probables.

Otro trabajo enfocado a esta tarea es el de Sannier et al. [16], en el cual proponen una taxonomía semántica y diferentes patrones a utilizar. Además, con estas herramientas, y con el apoyo del Framework GATE, crearon un clasificador el cual fue evaluado sobre textos legales en francés, más precisamente sobre textos legales de Luxemburgo y Canadá. Para el caso de estudio de los textos legislativos de Luxemburgo obtuvieron 87.57 % de medida F1, mientras que para el caso de estudio de Canadá obtuvieron un 80.59 %.

En Tuyen Le et al. [17] se puede encontrar detallado un estudio que aborda una problemática similar pero se enfoca en investigar la relación semántica entre términos relacionados a medios de transporte extraídos desde documentos de orientación técnica. Para esta tarea, utilizan como base Word Embeddings median-

te la herramienta Word2Vec, utilizando tanto CBOW como skip-gram como arquitecturas. Estos vectores luego son utilizados para la clasificación de los términos en conjuntos de sinónimos, hiperónimos y hipónimos tomando como referencia otros términos base. Este enfoque puede ser tenido en cuenta para clasificar semánticamente a una referencia según los vectores de palabras de su predicado.

Es de destacar que en el caso del ámbito legal, los trabajos investigados coinciden en que muchas veces es necesario, para una completa comprensión de la semántica, conocer información de la fuente legal a la cual pertenece el objeto legal referido. También mencionan que para obtener buenos resultados, es necesario poder contar con etiquetas lo más precisas posibles para el ámbito en el cual se está trabajando. La dificultad de esta tarea se deduce de la existencia de diferentes estudios que se enfocan específicamente en la definición de taxonomías para utilizar al momento de realizar una clasificación semántica. Un ejemplo de esto se puede ver en Maxwell et al. [18], donde se emplearon 32 horas hombre para analizar referencias en textos legales asociadas a restricciones sobre desarrollos de software. Otro punto a destacar es que no se encontraron trabajos de esta índole relacionados al área legal para el idioma español.

Capítulo 3

Base de Jurisprudencia Nacional

La Base de Jurisprudencia Nacional (BJN) es la principal fuente de información utilizada en este proyecto y consiste de un sistema de almacenamiento de sentencias judiciales de la Suprema Corte de Justicia, de Tribunales de Apelaciones y de Juzgados Letrados de Primera Instancia. La misma se creó con el objetivo no sólo de modernizar el manejo de la jurisprudencia de nuestro país si no de también poder brindarle una herramienta a la población para que pueda realizar diferentes búsquedas sobre la información judicial. Hoy en día cualquier persona es capaz de realizar consultas a la BJN desde su sitio web [6].

Número	Sede	Importancia	Tipo
309/2010	Tribunal Apelaciones Penal 1° T°	MEDIA	INTERLOCUTORIA

Fecha	Ficha	Procedimiento
20/09/2010	97-431/2009	PROCESO PENAL ORDINARIO

Materias
DERECHO PENAL

Firmantes	
Nombre	**Cargo**
Dr. Alberto Domingo REYES OEHNINGER	Ministro Trib.Apela.
Dra. Anabella Elizabeth DAMASCO SOLARI	Ministro Trib.Apela.

Redactores	
Nombre	**Cargo**
Dra. Anabella Elizabeth DAMASCO SOLARI	Ministro Trib.Apela.

Abstract	
Camino	**Descriptores Abstract**
DERECHO PENAL->DELITOS CONTRA LA PROPIEDAD MUEBLE E INMUEBLE->HURTO (ARTS. 340 A 342 DEL C.P.)	
DERECHO PROCESAL PENAL->MEDIOS DE IMPUGNACIÓN DE LAS RESOLUCIONES JUDICIALES->RECURSO DE APELACIÓN->APELACION INTERLOCUTORIA	

Figura 3.1: Información adicional de una Sentencia

Este sistema no sólo contiene las redacciones de las sentencias, si no que también contiene información adicional. Esta información se puede ver al realizar una consulta sobre la base desde su sitio web. En la figura 3.1 se puede ver la información que es proporcionada al usuario al momento de consultar una sentencia. Dentro de esta información se destaca la sede judicial emisora de la sentencia, la fecha de emisión, el identificador de la ficha o expediente asociada al caso y la materia a la cual aplica la sentencia, en el caso de la imagen "Derecho Penal". Otro detalle a destacar es el campo denominado "Camino", el cual hace referencia a una categorización realizada a la sentencia según el tema principal del caso.

Un punto importante sobre el sitio web que disponibiliza esta información es que le brinda a los usuarios la posibilidad de aplicar filtros para acotar la información a extraer. El usuario puede realizar dos tipos de consultas, una consulta simple o una consulta selectiva. La primera consiste en una búsqueda por texto, donde también se puede habilitar la opción de utilizar sinónimos, realizar una búsqueda de frase exacta o una búsqueda de algunas palabras. Por otro lado, la búsqueda selectiva ofrece al usuario otros filtros, como los que se pueden ver en la imagen 3.2. Dentro de los mismos se encuentran filtros relacionados a búsquedas de palabras claves como los campos "Resumen" y "Texto", filtros de fechas, y opciones de seleccionar entre los diferentes firmantes o redactores. Además, se encuentran otros filtros que son de utilidad a la hora de acotar el alcance de una búsqueda. Algunos de ellos son:

- Materia: filtro para determinar una única área sobre la cual buscar. Algunos ejemplos de materias son "Derecho Comercial" y "Derecho de Familia".
- Tipo: filtro que permite elegir entre los valores "Definitiva" e "Interlocutoria".
- Sedes: filtro que permite seleccionar un cojunto de sedes.
- Importancia: filtro que permite elegir qué nivel de importancia deben tener las sentencias a extraer. Los posibles niveles son "Alta", "Media" y "Baja".

En cuanto al filtro de "Importancia", cabe destacar que dicha clasificación es determinada al dar de alta la sentencia en la BJN. Los valores para el filtro "Tipo" hacen referencia al tipo de sentencia a buscar. Las sentencias definitivas son aquellas que resuelven el caso principal, mientras que las sentencias interlocutorias son aquellas que resuelven determinadas discusiones que surgen dentro del proceso y pueden incidir en la decisión final.

Algunos filtros cuentan con la opción de seleccionar con qué operador lógico van a ser utilizados. Estos operadores determinarán al final qué consulta se debe realizar sobre la base. Si se selecciona la opción "AND" en un filtro, los valores elegidos para el mismo deben estar presentes para que una sentencia sea recuperada y presentada al usuario. En los casos en los que se seleccione el operador "OR" se recuperarán aquellas sentencias que cumplan los valores especificados, o en caso de no cumplirlos, cumplan con las demás restricciones indicadas. Por último, si se selecciona el operador "NOT", se está indicando que los valores seleccionados para ese filtro no deben cumplirse en las sentencias que son recuperadas como resultado de la búsqueda.

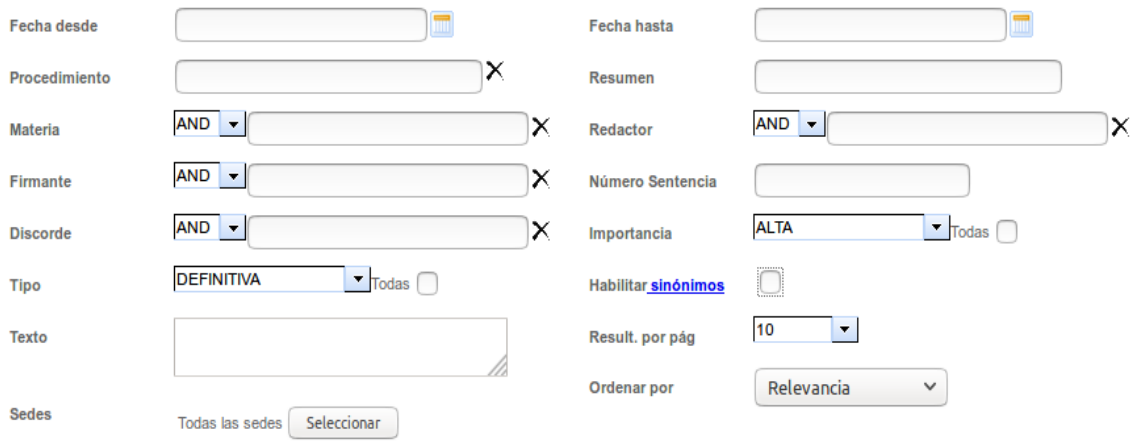

Figura 3.2: Filtros en búsqueda selectiva

En el contexto de este proyecto, se contó inicialmente con un subconjunto de información de la BJN que contiene las redacciones de las sentencias. Esta información está presente en formato CSV, donde cada registro contiene diferentes atributos dentro de los cuales el más importante es la redacción de la sentencia en formato HTML. Además de esta información, cada entrada contiene algunos datos explícitos de los que se muestran en la figura 3.1, como el número y año de la sentencia y el identificador del expediente del caso. También cada registro contiene información parcial sobre otras propiedades de la sentencia, como por ejemplo el identificador de la sede. Esta información por si sola no es de mucha utilidad ya que se trata de un identificador numérico de manejo interno del sistema de la BJN. Sin embargo, en el transcurso del proyecto se pudo contar con información que permitió relacionar al identificador de sede presente en cada registro del archivo CSV principal y la información de la sede correspondiente a dicho identificador. La información inicial con la cual se contó consistía de 72358 sentencias cuyos registros abarcan cerca de 1.8GB.

En cuanto al texto de las sentencias almacenadas en la BJN, se puede destacar principalmente su estructura, la cual está definida por diferentes secciones en el siguiente orden: “Vistos”, “Resultando”, “Considerando” y “Falla” o “Resuelve”. La primera sección se corresponde a una introducción a la problemática tratada en el caso, la sección “Resultando” realiza un resumen de lo sucedido en la Sentencia de Primera Instancia, la sección “Considerando” es en la que se encuentran los argumentos en los que se basa el Tribunal para fallar en la Sentencia de Segunda Instancia, y por último la sección “Falla” o “Resuelve” es el resultado de la Sentencia de Segunda Instancia.

Capítulo 4

Expresiones Regulares y generación del Corpus

La primera etapa de este proyecto consistió en realizar un análisis de las sentencias presentes en la BJN, y a partir de la misma generar un sub-corpus anotado de entrenamiento para las etapas posteriores. El primer paso para esto fue analizar distintas sentencias y reconocer qué objetos legales eran los que poseían mayor cantidad de referencias, y cuáles de estos tenían mayor influencia sobre el desenlace del caso. Culminado el análisis sobre un conjunto de sentencias, se definió el conjunto de objetos legales sobre los cuales poner énfasis a la hora de extraer referencias. Este conjunto está compuesto por:

- Sentencias
- Artículos
- Leyes
- Decretos
- Fichas o Expedientes
- Doctrinos

En el proceso de definición de este conjunto también se planteó la posibilidad de incluir a las referencias a Fojas, debido a que este tipo de referencias aparece en grandes cantidades a lo largo de cada sentencia y se supuso referían al expediente principal del caso. Sin embargo, luego de diferentes consultas realizadas a personal del Poder Judicial, se decidió no incluir a la Foja como un objeto de interés ya que las referencias a las mismas no aportarían mucha información al usuario que lee la sentencia, debido a que por lo general, este no cuenta con el expediente del caso.

Culminada la etapa de definición de los objetos relevantes, y basándose en la literatura investigada sobre proyectos similares, se decidió utilizar en primera instancia expresiones regulares para la extracción de referencias de los objetos legales antes mencionados sin incluir a los Doctrinos, los cuales se trabajaron con

un enfoque diferente. Si bien el primer objetivo era utilizar las expresiones regulares como herramienta de extracción de referencias, debido a la complejidad de las mismas y a la carencia de esta técnica para abarcar la totalidad de las referencias existentes, se decidió utilizarlas como una herramienta para generar un sub-corpus a ser utilizado en métodos de aprendizaje automático. En las siguientes secciones se detalla tanto el proceso de creación de las expresiones como el formato elegido para guardar los resultados y el método utilizado para evaluar la correctitud de las mismas. Además se hace mención al proceso de generación del sub-corpus y la composición del mismo.

4.1. Expresiones Regulares

A pesar de que existen formatos definidos para la escritura de referencias a objetos legales, estos no son siempre respetados. Sumado esto a la gran cantidad de sentencias con las que cuenta la BJN, se pudieron identificar diversas formas de expresar las referencias a los objetos antes mencionados. Es por este motivo que se decidió destinar un conjunto de 100 sentencias aleatorias para el proceso de creación de las expresiones regulares, con el objetivo de que estas sean lo más generales y correctas posibles para abarcar la mayor cantidad de referencias existentes en todo el conjunto de sentencias.

4.1.1. Proceso de creación

Para el proceso de creación se utilizó el módulo `re` de Python [19], el cual permite crear y trabajar con expresiones regulares y cadenas de texto. Se desarrolló un módulo encargado de aplicar las expresiones regulares a cada sentencia y almacenar los resultados para su posterior evaluación. Debido a que en casos como los de los Artículos resultaba imposible crear una única expresión regular que satisficiera las diferentes formas con las que se referenciaban, se procedió a crear reglas para los distintos casos reconocidos. Luego se ordenaron las mismas según su especificidad para así poder ejecutarlas en orden con el objetivo de que al momento de coincidir con una regla más específica, se corte la ejecución de las reglas menos específicas.

En la figura 4.1 se puede observar un ejemplo de las reglas utilizadas para extraer referencias a artículos ordenadas desde la más específica a la más general. Estas se diferencian en el orden en el cual se espera que aparezcan las distintas variables en el texto, por ejemplo la regla `EANS` reconocerá a aquella referencia que comience con elementos como inciso, numeral o literal (variable `extra`), esté seguida de una variante de la palabra artículo (por ejemplo art., arts., artículos) junto a los números del o de los artículos referidos (incluyendo incisos, literales y numerales) y finalice con la fuente legal a la cual pertenece el artículo.

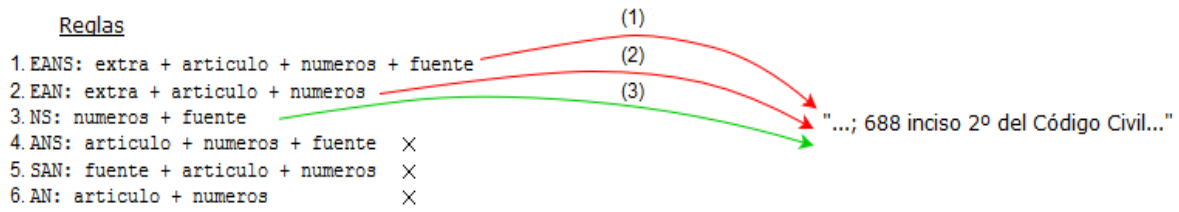

Figura 4.1: Ejemplo de aplicación de reglas

También en la figura 4.1 se ilustra el proceso de aplicación de las reglas, donde se puede ver que las primeras dos reglas fallan debido a que la referencia no contiene ningún inciso, literal o numeral antes del número del artículo; así como tampoco se encuentra una variante de esta palabra. Luego de ejecutar las reglas 1 y 2, se busca una coincidencia con la regla 3, la cual resulta exitosa, y de esta forma evita que se busquen coincidencias para las reglas 4, 5 y 6. Para lograr este comportamiento se utilizó el método `finditer`, el cual retorna todas las coincidencias que no se superponen en forma de iterador de `Match Objects`, los cuales contienen información relevante como por ejemplo, qué grupos arrojaron resultados, y los caracteres del string entre los cuales se reconoció el patrón.

"...Y art. 1955 del C. Civil, el Tribunal falla..."

```
Coincidencia completa: 7369-7391: art. 1955 del C. Civil
Coincidencia grupo 1: 7374-7379: 1955
Coincidencia grupo 2: 7384-7391: C. Civil
```

Figura 4.2: Ejemplo de coincidencias

Como se puede ver en la figura 4.2, en la salida del programa se obtiene una coincidencia para un artículo entre los caracteres 7369 y 7391 de una oración, y a su vez se obtienen coincidencias para dos grupos, de los cuales el primero es el número del artículo y el segundo es la fuente legal a la que pertenece el mismo.

En cuanto a las expresiones regulares creadas para los objetos legales restantes, en algunos casos también fue necesario separar en reglas, por ejemplo para diferenciar casos en los cuales la referencia se hacía sobre más de una instancia del mismo objeto (plural), o sobre una única instancia (singular), y realizar algunas suposiciones como en el caso de las referencias a sentencias, ya que en la mayoría de los ejemplos la referencia comienza con la palabra "sentencia" o alguna de sus derivadas, por lo que basta con buscar de ahí en adelante. De esta manera, se simplifican las reglas y se evitan demoras excesivas a la hora de procesar las sentencias debido al backtracking generado por las expresiones regulares.

Para evitar que una modificación en una regla afectara los resultados obtenidos anteriormente para algunas sentencias, se creó un módulo que compara el resultado de coincidencias utilizando las reglas modificadas, con el resultado obtenido con la última versión de las reglas, generando una alerta al detectarse cambios. De esta forma sólo basta con observar los cambios en las sentencias que generaron alerta para verificar que no se haya afectado el funcionamiento anterior de las reglas. Para lograr este comportamiento se utilizó el módulo `diff` de Python, el cual recibe como parámetros dos cadenas de texto y verifica si existen diferencias, retornando las mismas. Este subproceso de generación y validación de expresiones regulares puede verse ilustrado en imagen 4.3.

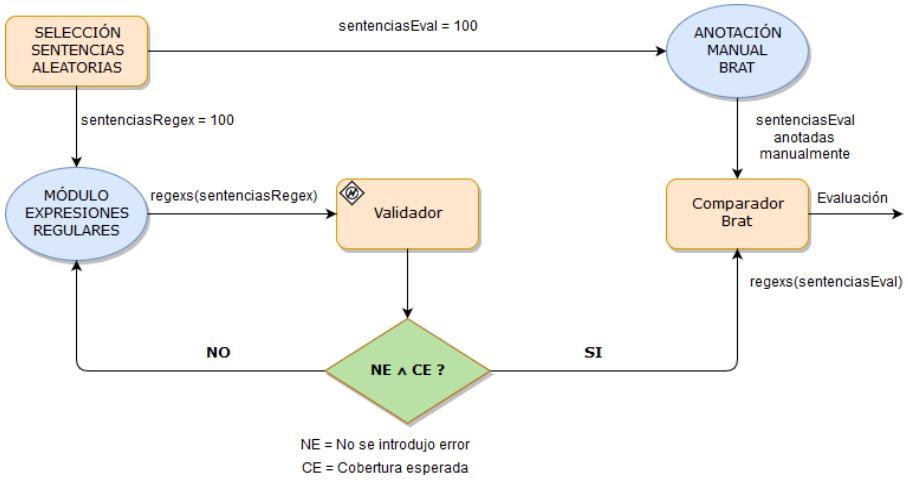

Figura 4.3: Diagrama del proceso de extracción

4.1.2. Proceso de anotación

Luego de finalizada la etapa de creación de las expresiones regulares, se procedió a evaluar las mismas tomando de manera aleatoria un conjunto de 100 sentencias diferentes a las consideradas anteriormente. Con el objetivo de poder realizar una evaluación de mejor calidad, se decidió anotar manualmente este conjunto de sentencias. Si bien es una cantidad reducida comparada con las casi 72.000 sentencias disponibles, el tiempo dedicado a anotar manualmente dichas sentencias fue muy grande, por lo que se hubiera hecho inviable intentar considerar un conjunto de mayor tamaño.

Para el proceso de anotación manual se utilizó la herramienta BRAT [20], la cual presenta una interfaz web intuitiva y amigable para anotar documentos en texto plano. Para poder utilizar esta herramienta se debe definir antes un esquema de anotación[1], el cual puede contener elementos como entidades, relaciones entre las mismas, eventos y atributos. BRAT presenta un proceso de anotación bastante simple, en el cual el usuario selecciona un archivo de texto a anotar, y luego simplemente seleccionando la zona de texto deseada, la herramienta despliega las diferentes opciones de entidades existentes en su esquema para que el usuario seleccione la que desea, y además asigne algún atributo si corresponde. Una vez que se anotan las entidades, el usuario puede crear una relación simplemente arrastrando una entidad hacia la otra. Luego de dicha acción, BRAT despliega las relaciones existentes en el esquema de anotación, que admitan dos entidades de tipo igual al de las seleccionadas.

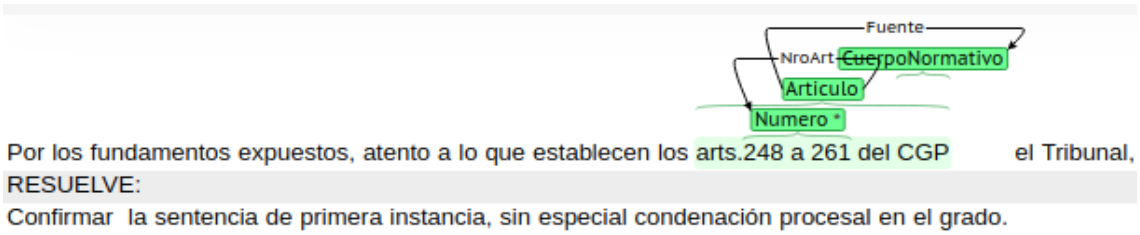

Figura 4.4: Ejemplo de esquema de anotación para Artículos

[1]Ver esquema de anotación en Anexo A

La figura 4.4 muestra cómo el usuario ve las anotaciones que está realizando. Como se puede ver en dicho ejemplo, además de entidades, se incluyeron elementos como relaciones y atributos con el objetivo de facilitar el procesamiento en etapas posteriores. Esto se debe a que contar con la información de si un grupo contiene uno o más números permite mejorar la calidad de los datos extraídos, lo cual a su vez permite reducir el ruido al momento de utilizar los mismos.

Luego, las anotaciones realizadas sobre el texto, son almacenadas por la herramienta en archivos de extensión `.ann` con un formato definido por BRAT. Dicho formato se ve reflejado en el ejemplo de la figura 4.5, donde se ve cada anotación en una línea diferente, la cual contiene un identificador del elemento anotado que a su vez distingue entre entidad o relación, como es el caso de T38 para la entidad artículo comprendida por "*arts.248 a 261 del CGP*", el nombre de la entidad, relación o atributo anotado, el rango de caracteres del texto en el cual se realizó la anotación y por último el texto anotado.

```
T38 Articulo 9226 9248 arts.248 a 261 del CGP
T39 Numero 9231 9240 248 a 261
A1 Varios T39
T40 CuerpoNormativo 9245 9248 CGP
R25 Fuente Arg1:T38 Arg2:T40
R26 NroArt Arg1:T38 Arg2:T39
```

Figura 4.5: Ejemplo salida BRAT

Luego de anotar las 100 sentencias seleccionadas para la evaluación, siguiendo el esquema presentado en la figura 4.3, se procedió a aplicarle las expresiones regulares y almacenar sólo los resultados de las coincidencias completas (sin tener en cuenta los grupos) con el formato BRAT mencionado anteriormente. De esta forma, se pudo utilizar la funcionalidad de comparación que presenta esta herramienta, la cual consiste en presentar en la interfaz las dos versiones de la sentencias anotadas, una resultante de la anotación manual y la otra resultante de aplicar las expresiones regulares.

4.1.3. Evaluación y Resultados

Luego de obtener las dos versiones de las sentencias anotadas, el proceso de comparación fue manual, comparando referencia a referencia y clasificando la extracción en uno de los siguientes cuatro estados: Verdadero Positivo, Falso Positivo, Falso Negativo o Incompleto. A continuación se detallan los criterios para asignar cada uno de estos estados:

- Verdadero Positivo: se asignó a aquellos casos en los que la referencia extraída mediante expresiones regulares coincidía en un 100 % con la anotada manualmente.
- Falso Positivo: se asignó en los casos en que las expresiones regulares identificaron una sección de texto la cual no presentaba una referencia a ninguno de los objetos legales considerados.
- Falso Negativo: fue asignado en los casos en los que una referencia anotada manualmente no fue reconocida por las expresiones regulares.

- Incompleto: se asignó en los casos en los que la referencia extraída coincidía parcialmente con la anotada manualmente.

Como muestra la tabla 4.1, la mayor parte de las referencias con estado Incompleto se dieron en los artículos debido a que al momento de crear las expresiones regulares se utilizó un conjunto de cuerpos normativos conocidos, el cual no contenía a muchos otros existentes, como por ejemplo el *"Código de la Niñez y Adolescencia"*. Debido a la existencia de casos incompletos, surgieron dos formas de interpretar los resultados. La primera consiste en evaluar las expresiones regulares tomando a los casos con estado Incompleto como Falsos Negativos, es decir tener en cuenta como correctas sólo a las referencias extraídas con un 100 % de exactitud. La otra forma consiste en tener en cuenta a los Incompletos como Verdaderos Positivos, debido a que son casos en los que se reconoció la existencia de una referencia pero falta información para poder identificarla completamente. Un ejemplo de esto se dio con la referencia "*artículo 72 del C. de la Niñez y Adolescencia*", sobre la cual las expresiones regulares extrajeron el número pero no la fuente.

	Artículo	Sentencia	Ley	Decreto	Ficha
VP	733	253	195	38	109
FP	1	8	1	1	0
FN	26	13	10	7	1
INC	60	8	0	7	0

Tabla 4.1: Resultados expresiones regulares

En cuanto a las métricas utilizadas para evaluar tanto este método de extracción de referencias como los siguientes, se tomó en cuenta lo mencionado en Konkol [12]. El autor destaca que en los problemas de NER, en comparación con otras tareas de PLN, utiliza únicamente un método estándar de medidas conformado por Precision, Recall y medida F1. La Precision **(P)** se calcula como P = VP/(VP+FP), dando cuenta de qué tan precisa es la expresión regular en cuanto a la extracción de referencias, es decir qué proporción de las secuencias de texto marcadas como referencias por las expresiones regulares son efectivamente referencias válidas. El Recall **(R)**, también conocido como Recuperación, se calcula como R = VP/(VP+FN), y refleja qué tan bien extrajo la expresión regular las referencias existentes en las sentencias, es decir qué proporción de las referencias existentes fueron reconocidas correctamente por las expresiones regulares. Por último, la medida F1 **(medida F)** es la media armónica entre las dos medidas antes mencionadas (MEDIDA F = 2P.R/(P+R)).

Como se puede observar en las tablas 4.2 y 4.3, basándose en las expresiones regulares para la extracción de referencias se obtuvieron muy buenos resultados, notándose poca diferencia entre las dos formas de tomar a las extracciones incompletas debido a que la cantidad de estas fue pequeña para la mayoría de los casos. En el caso en el que se puede ver mayor diferencia es en el de los decretos, ya que entre FN e INC abarcan el 26 % de los ejemplos totales, aportando las referencias de clase INC el 50 % de las referencias finalmente catalogadas como FN en el caso de la tabla 4.2, motivo por el cual la medida Recall aumenta para este caso.

	Precision	Recall	F1-Score
Articulo	99.86	89.50	94.39
Sentencia	96.93	92.34	94.58
Ley	99.49	95.12	97.26
Decreto	97.44	73.08	83.52
Ficha	100	99.09	99.54

Tabla 4.2: Incompletas como FN

	Precision	Recall	F1-Score
Articulo	99.87	96.83	98.33
Sentencia	97.03	95.26	96.14
Ley	99.49	95.12	97.26
Decreto	97.82	86.54	91.84
Ficha	100	99.09	99.54

Tabla 4.3: Incompletas como VP

A continuación se presentan algunos ejemplos de casos en los cuales las expresiones regulares fallaron:

- *"...Decreto P.E. No. 851/76..."*: este caso no fue reconocido por las reglas desarrolladas para extraer referencias a decretos debido a que en el conjunto de sentencias utilizadas como apoyo en el proceso de creación, en ningún momento se observó una secuencia similar a "P.E.", sigla que hace referencia al Poder Ejecutivo. Por esta razón nunca se tuvo en consideración que entre la variante de la palabra decreto y el número pudiera existir otra secuencia de texto.
- *"...art. 46 CP, nums. 7º y 13 respectivamente..."*: en este caso la extracción fue incompleta, ya que no se obtuvo la información de los numerales debido a que no se implementó ninguna regla que considerara que elementos como incisos, literales y numerales pudieran ocurrir luego de la aparición de la fuente legal. Es por esto, que esta secuencia de texto coincidió con la regla que espera una variante de la palabra artículo, uno o varios números junto a incisos, literales y numerales, y por último el cuerpo normativo o fuente legal a la que pertenece el/los artículos referidos.
- *"...a. 61 de la L. 15.750..."*: este caso resulta interesante, debido a que no se reconoció ni la referencia al artículo, ni tampoco la referencia a la ley. Esto se debe, como en los casos anteriores, a que al momento de crear las expresiones regulares, no apareció ningún caso en el que la palabra "artículo" se abreviara como "a.", y tampoco ningún caso en el que la palabra "ley" se abreviara como "L.".

Si bien el proceso de creación de las expresiones regulares resultó difícil y requirió de mucho tiempo, cabe destacar que se obtuvieron resultados mejores de lo esperado. Sin embargo, es necesario hacer énfasis en que la evaluación de estas expresiones regulares se realizó sobre un conjunto muy acotado de sentencias, pudiendo existir en las restantes, formatos de escritura de referencias muy diferentes a los vistos en dicho conjunto como por ejemplo los analizados recientemente. Es por esto que surgió la necesidad de experimentar con otros métodos de extracción de referencias más escalables, que fueran capaces de realizar generalizaciones sobre los datos permitiendo así obtener buenos resultados sobre todo el conjunto de sentencias. En consecuencia, y como se mencionó anteriormente, las expresiones regulares pasaron a ser una herramienta a utilizar en la generación de sub-corpus para métodos posteriores.

4.2. Generación del sub-corpus

Una vez tomada la decisión de experimentar con métodos de aprendizaje automático para abordar la tarea de extracción de referencias, fue necesario generar un corpus anotado para el entrenamiento y posterior evaluación. Si bien, se contaba con 100 sentencias anotadas, esto resultaba muy poco para el correcto entrenamiento de los modelos. Por otro lado, realizar la anotación manual de una cantidad grande de sentencias conllevaba un tiempo extremadamente largo, razón por la cual se utilizaron las expresiones regulares como base para la anotación. Para esto se tomaron 1000 sentencias aleatorias de la BJN y se procedió a aplicarle el sistema de reglas manuales desarrollado mediante expresiones regulares con el objetivo de generar un conjunto de entrenamiento ya anotado con formato BRAT. Como se mencionó en la sección 4.1, el sistema basado en reglas presenta problemas para algunas sentencias, debido a que algunas reglas son demasiado generales y provocan que se caiga en un proceso de backtracking muy costoso. Es por esto que al momento de aplicar este sistema a las 1000 sentencias se configuró un temporizador para que una sentencia no detenga la ejecución. Es decir, se esperó un tiempo razonable a que la sentencia fuera anotada, y se la descartó si este tiempo era superado. Una vez culminada la anotación automática de las sentencias, se revisaron manualmente estas anotaciones corrigiendo los errores cometidos por las expresiones regulares y anotando aquellas sentencias descartadas en primera instancia.

Luego de este proceso, el siguiente paso consistió en realizar una separación 80/20 de este conjunto destinando el 20 % al conjunto de evaluación, mientras que el 80 % fue a su vez dividido de la misma forma en un conjunto de entrenamiento y un conjunto de validación. Por lo tanto, se destinaron 200 sentencias al conjunto de evaluación, 640 al conjunto de entrenamiento y las restantes 160 al conjunto de validación. En la siguiente tabla se pueden ver los totales de referencias contenidas en cada conjunto.

	Artículo	Sentencia	Decreto	Ley	Ficha	**Total**
Entrenamiento	5709	2406	444	1361	846	10766
Validación	1447	569	94	443	239	2792
Evaluación	1563	736	111	481	242	3133
Total	8719	3711	649	2285	1327	16691

Tabla 4.4: Cantidad de referencias por conjunto

Capítulo 5

Extracción de referencias utilizando Aprendizaje Automático

En esta sección se documentan los detalles del proceso de extracción automática de referencias mediante la utilización de métodos de aprendizaje supervisado. En todos los casos, para el entrenamiento y la evaluación se utilizó el sub-corpus presentado en el capítulo anterior. Cabe destacar que la mayoría de las herramientas de NER no soportan anotaciones superpuestas, por lo que para esta sección se tuvieron que filtrar las anotaciones para contar únicamente con aquellas anotaciones a entidades completas. Es decir, se dejaron afuera las anotaciones de las propiedades, y por lo tanto estas fueron extraídas mediante otros procesos.

Un concepto importante que es necesario para comprender el funcionamiento y la utilización de estas herramientas, es el de "Modelado Secuencial". Este concepto surge de la necesidad de abordar problemas de procesamiento de secuencias de datos de distintos tipos como por ejemplo de audio o de texto. Un problema para el cual se utiliza el modelado secuencial es el de la traducción de textos o el de Reconocimiento de Entidades Nombradas (NER). Estos modelos se basan en el principio de que un estado en particular depende del estado anterior. Las Redes Neuronales Recurrentes son muy utilizadas en estos problemas, debido a que presentan bucles en su arquitectura, lo que permite que la activación de cada neurona, no dependa únicamente de la entrada actual, si no también de los valores de activación previos.

En el caso de esta etapa, el problema a resolver fue catalogado de tipo NER, debido a que es necesario etiquetar secuencias de texto con una de varias clases definidas (Artículo, Sentencia, Decreto, Ley y Ficha). Para esto, se debe definir si una palabra pertenece a una clase o no, y esto depende no sólo de la palabra si no que también de su contexto. Es por estas razones, que esta tarea está incluida dentro del área del "Etiquetado de Secuencias" (Sequence Labeling). En los problemas de este tipo, generalmente se le asigna una etiqueta a cada token, la cual indica si el mismo pertenece a una entidad nombrada o no. Es decir, en el texto "como indica el artículo 200 del C.G.P", el objetivo es etiquetar a los tokens "como", "indica", y "el" como no pertenecientes a una entidad nombrada, y a los tokens restantes como pertenecientes a una entidad nombrada de clase Artículo. Para esto se utilizan distintos formatos de etiquetas, dentro de los cuales se encuentra el formato BIO (Begin-Inside-Outside). Este formato utiliza tres etiquetas principales:

- **B-X**: asignada al token que da comienzo a una entidad nombrada de clase X. En el ejemplo presentado anteriormente, al token "artículo" se le asignaría la etiqueta B-Artículo.
- **I-X**: asignada a los tokens que componen una entidad nombrada de clase X, sin contar al primer token (etiquetado como B). En el ejemplo los tokens "200", "del", "C.G.P" serían etiquetados con I-Artículo.
- **O**: asignada a todo aquel token que no pertenezca a una entidad nombrada de cualquier clase.

5.1. NeuroNER

El primer enfoque utilizado para la extracción automática de referencias a objetos legales fue NeuroNER [21]. Este es un sistema de Reconocimiento de Entidades Nombradas desarrollado en Python utilizando TensorFlow [22], el cual obtiene resultados similares a los del estado del arte al evaluarse con las entidades comúnmente utilizadas como lo son organizaciones, nombres de personas y locaciones. Este sistema fue incluido en el proceso de extracción con la finalidad de poder entrenarlo tomando a los ejemplos de los diferentes objetos legales como entidades nombradas, para así obtener un modelo que posea una capacidad de generalización mayor al de las expresiones regulares. A continuación se presentan detalles del sistema y de cómo éste fue utilizado para la extracción de referencias a objetos legales.

5.1.1. Arquitectura

NeuroNER está basado en una variante de las redes neuronales recurrentes, denominadas Long Short Term Memory Bidireccionales (Bi-LSTM), utilizada junto con representaciones vectoriales de las palabras (Word Embeddings) mejoradas con representaciones vectoriales de caracteres (Character Embeddings) y CRF (Campos Aleatorios Condicionales) [23].

Más precisamente, la arquitectura de este sistema consiste en tres capas principales. La primera capa toma como entrada los vectores generados por el sistema para cada caracter y los utiliza como entrada en dos redes LSTM (una procesa los datos desde la primera letra hasta la última y la segunda al revés). El objetivo de esto es mitigar los problemas que presenta la utilización de los vectores de palabras, debido a que los vectores de palabras desconocidas deben ser inferidos a partir del contexto en el cual aparece la misma. Con la utilización de los vectores por caracter, se aprovecha la información que está presente en los prefijos y sufijos de las palabras. La siguiente capa toma como entrada las salidas de las redes LSTM anteriores y los vectores de cada palabra, los cuales pueden estar generados por el sistema a partir de una inicialización aleatoria o a partir de vectores pre-entrenados, o ser sólo vectores pre-entrenados, y los utiliza otra vez como entradas en otras dos redes LSTM de la misma manera que para los caracteres. En la última capa, se utilizan los resultados de estas dos LSTM como entradas de una capa CRF (opcional), la cual genera vectores de probabilidad combinando las diferentes etiquetas obtenidas por las LSTM y sus probabilidades. Finalmente se asignan las etiquetas a cada palabra según el vector con mayor probabilidad.

Cabe destacar que la capa CRF tiene la capacidad de aprender restricciones de los datos de entrenamiento, las cuales son útiles para disminuir la cantidad de secuencias de etiquetas no válidas a considerar. Entre estas restricciones se encuentran:

- La primera palabra de una oración debe comenzar con la etiqueta "B" u "O", no con "I".
- En la secuencia de etiquetas "B-etiqueta1 I-etiqueta2 I-etiqueta3 I- ...", etiqueta1, etiqueta2, etiqueta3... deben ser del mismo tipo de etiqueta. Por ejemplo, en el contexto de las sentencias, "B-Articulo I-Articulo" es válido, pero "B-Articulo I-Sentencia" no.
- "O I-etiqueta" no es válido. La primera etiqueta de una entidad nombrada debe comenzar con "B" y no con "I", en otras palabras, el patrón válido debe ser "O B-etiqueta".

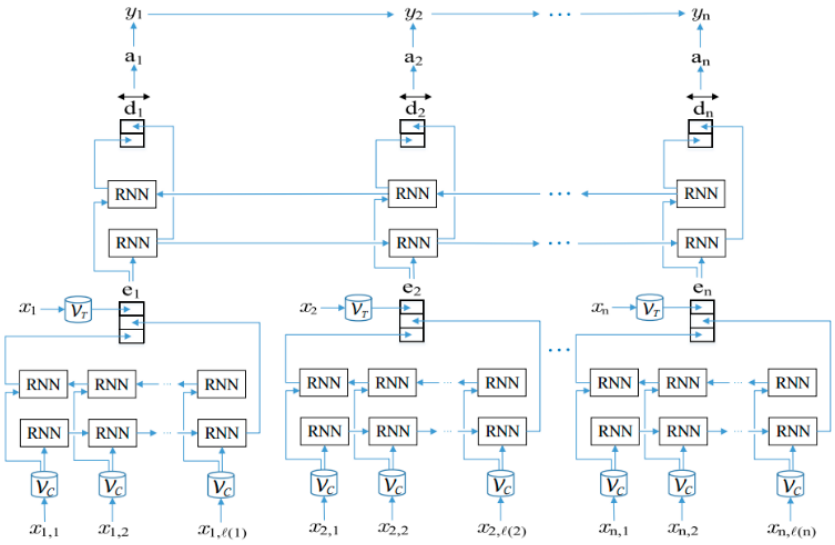

Figura 5.1: Arquitectura NeuroNER. *Extraído de www.neuroner.com [3]*

En la figura 5.1 se puede apreciar en detalle la arquitectura comentada. Entre los principales parámetros se encuentran n el cual representa el número de palabras a procesar y Vt el cual corresponde al mapeo entre la palabra y su vector. A su vez, como se comentó anteriormente, cada palabra es además procesada por caracter, donde xi,j se corresponde al caracter j de la palabra i por lo que $xi,l(i)$ se corresponde al último caracter de la palabra i. Así como se realiza un mapeo entre una palabra y su vector, también se realiza un mapeo entre un caracter y su vector. Este mapeo se puede apreciar en la figura con el símbolo Vc. Luego, $\mathbf{e}_i$ es el vector mejorado de la palabra i, $\mathbf{d}_i$ es la salida de la capa de predicción de etiquetas por palabra, $\mathbf{a}_i$ es el vector de probabilidades de la palabra i sobre las etiquetas disponibles e y_i es la etiqueta finalmente asignada al token i.

5.1.2. Características principales

Facilidad de uso

Una de las características principales del sistema es su facilidad de uso y su gran capacidad de adaptación para conjuntos de datos de diferentes temáticas. Esto se debe a que está basado en redes neuronales siendo capaz de ajustar los parámetros del modelo a medida que analiza el conjunto de datos. Esto le permite al usuario ahorrarse el trabajo de realizar una selección manual de atributos sobre las cuales trabajar, lo cual muchas veces depende de los datos que se vayan a utilizar y por lo tanto requiere un re-trabajo cuando se quiere reutilizar en un área diferente.

Otra característica importante, y una de las principales razones por las cuales se eligió esta herramienta es que, a pesar de que NeuroNER trabaja con los conjuntos de datos en formato CoNLL [24] con etiquetado BIO o BIOES, también brinda soporte para conjuntos de datos anotados en formato BRAT realizando la transformación entre este formato y el formato CoNLL. De esta forma fue posible reutilizar el trabajo realizado con la herramienta BRAT y el conjunto de reglas manuales desarrolladas mediante expresiones regulares del cual se habló en la sección anterior.

Resultados en tiempo real

Una gran cualidad de este sistema es que brinda al usuario los resultados de evaluar el conjunto de evaluación luego de cada iteración, almacenando además el modelo resultante para que el usuario al final pueda elegir, según sus propios criterios, aquel que más se adecúe al objetivo buscado. Este aspecto resulta beneficioso para experimentos con grandes cantidades de datos debido a que generalmente en dichos casos el entrenamiento puede llevar muchas horas.

Dentro de los resultados generados se pueden ver matrices de confusión, medidas de Precision, Recall y F1. Además la herramienta genera la salida anotada de cada conjunto de datos para que se pueda analizar su comportamiento a medida que transcurre el proceso de entrenamiento. A su vez, las matrices de confusión y las medidas de evaluación se presentan en cuatro formas diferentes: evaluación BIO, evaluación binaria, evaluación CoNLL y evaluación por token.

Formas de Evaluación

Las formas de evaluación utilizadas por NeuroNER se diferencian en la unidad de comparación utilizada. Es decir, algunas consisten en realizar comparaciones token a token, mientras que otras comparan a la entidad como un todo (secuencia de tokens). Como ejemplos de la primera se tienen a las evaluaciones por BIO y por token. La diferencia es que en la evaluación BIO, un verdadero positivo está dado únicamente en el caso en el que la etiqueta predicha coincide con la etiqueta anotada. Como se puede ver en la tabla 5.1, para los primeros 5 tokens se obtendrán verdaderos positivos, mientras que para el sexto no. Otro caso posible se daría si el modelo asignara la etiqueta B-Sentencia al token “T.A.C.”, caso en el cual la evaluación BIO no reconocería un verdadero positivo. Por otro lado, este caso sería tomado como verdadero positivo en la evaluación denominada “por token”, ya que esta forma de evaluación toma a las clases B e I como una sola.

Token	Sentencia	Inicio	Fin	Etiqueta real	Etiqueta predicha
sentencia	100	1566	1575	B-Sentencia	B-Sentencia
Nº	100	1576	1578	I-Sentencia	I-Sentencia
179/2007	100	1579	1587	I-Sentencia	I-Sentencia
del	100	1588	1591	I-Sentencia	I-Sentencia
T.A.C.	100	1592	1598	I-Sentencia	I-Sentencia
1º	100	1599	1601	I-Sentencia	O

Tabla 5.1: Ejemplo formas de evaluación

En cuanto a las formas de evaluación basadas en la comparación entre entidades, se utiliza la evaluación CoNLL y la evaluación binaria. En el primer caso, un verdadero positivo se dará cuando una determinada secuencia de tokens clasificadas por el modelo con una determinada etiqueta coincida totalmente, es decir en los tokens y sus etiquetas, con la entidad anotada. En el ejemplo de la tabla 5.1 se puede observar que para la evaluación CoNLL este no es un ejemplo positivo debido a que para el modelo la entidad nombrada es *"sentencia Nº 179/2007 del T.A.C"* cuando en realidad abarca hasta el token "1º". Por otro lado, la evaluación binaria no tiene en cuenta la etiqueta predicha por el modelo, si no que únicamente evalúa si la secuencia de tokens reconocida como una entidad, coincide totalmente con una entidad anotada. Un ejemplo de verdadero positivo en este caso se daría cuando una determinada secuencia de tokens es reconocida como una entidad (es decir una etiqueta B seguida de etiquetas I) pero su clase difiere de la anotada, como podría pasar si el modelo asignara la clase Articulo a las etiquetas de la columna "Etiqueta real" del ejemplo 5.1.

Para este trabajo se tomó como principal a la medida CoNLL, debido a que es la más específica y en este caso es de interés obtener las referencias completas, ya que la falta de un token puede llevar posteriormente a una mala interpretación y procesamiento de los datos.

5.1.3. Utilización

Como se mencionó en los puntos anteriores, NeuroNER brinda soporte para aquellos conjuntos de datos que están anotados en formato BRAT. Esto resultó útil en el marco de este trabajo ya que dicha herramienta fue utilizada en el proceso de anotación para la evaluación de las expresiones regulares, que luego dieron lugar a la generación del sub-corpus. Por lo tanto, al ya contar con un sub-corpus anotado en formato BRAT, resultó conveniente comenzar por una herramienta que soporte dicho formato y ahorre el trabajo de transformar los conjuntos de entrenamiento.

5.1.4. Separación de oraciones

Una de las tareas que realiza NeuroNER como parte del pre procesamiento del conjunto de datos, es la de separar los textos en oraciones y a su vez separar estas oraciones en tokens. Debido a que la correctitud de este proceso es de gran importancia para los resultados, se investigó acerca de las herramientas utilizadas para cumplir dicha tarea. Por defecto NeuroNER ofrece la posibilidad de utilizar los reconocedores

de oraciones y tokenizadores de SpaCy [25] o Stanford [26]. Sin embargo, al ser una herramienta de código abierto, permite también editar el código fuente para poder incluir alguna de las otras API's que brindan métodos similares para realizar las mismas tareas. Por estas razones se decidió evaluar diferentes herramientas haciendo énfasis en qué tan bien estas reconocían oraciones. Las herramientas seleccionadas para esto fueron las mencionadas anteriormente soportadas por defecto en NeuroNER, y la herramienta de Freeling [27].

Para evaluar el comportamiento de las mismas se seleccionaron un conjunto de fragmentos de texto pertenecientes a distintas sentencias y se procedió a comparar la salida de las herramientas seleccionadas, siempre comparando con la correcta separación en oraciones del texto.

> *"El agravio propuesto con relación a gastos por cambio de motor, no puede ser recepcionado.* 1 2 3 *Por sencilla aplicación subespecie del principio de congruencia vigente (**art.** 3 **198 C.G.P.**), cuando se compulsa que la pretensión referida al extremo (fs.* 2 3 *23-29 especialmente lit.* 2 3 *c. en fs.* 2 3 *26-27vto.* 2 3 *y pet.* 2 3 *2 en fs.* 2 3 *29), no resultó incluida en el objeto del proceso delimitado en audiencia del 5/12/2002 (fs.* 2 3 *52), en solución "citra-petita" que no resultó puntualmente cuestionada por los interesados, por lo que adquiere en la actualidad eficacia integral de cosa juzgada (**arts.** 2 3 **215 y conc.** 2 3 **C.G.P.**).* 1 2 3 *Abundantemente, corresponde consignar, que cuando los gastos para adquisición-instalación se verificaron con anterioridad a la ocurrencia del evento dañoso (ver fs.* 2 3 *8-11), los pretendidos no pueden entenderse causalmente conectados con éste y por consiguiente en calidad de daño emergente resarcible, y en todo caso, lo que pudiera eventualmente haberse reclamado, era el necesario costo de extracción y/o eventuales detrimentos en punto a la nueva utilización del implemento, lo que sublite no aconteció.* 1 2 3 *"*

Figura 5.2: Fragmento de texto utilizado en la evaluación

En la figura 5.2 se marcaron los límites de oraciones que determinaron las herramientas utilizadas. La referencia 1 indica el fin de una oración utilizando la herramienta Freeling, la referencia 2 indica el fin de una oración utilizando la herramienta de Stanford mientras que la referencia 3 refiere al fin de una oración indicada por SpaCy. Como se puede observar, Freeling tuvo el mejor comportamiento ya que reconoció correctamente tres oraciones. Por otro lado, SpaCy y Stanford coinciden en la mayoría de los casos, presentando problemas para reconocer abreviaciones como por ejemplo *"fs. 26-27 vto."*. Visto esto, con el objetivo de comparar dos enfoques diferentes, se decidió utilizar SpaCy y Freeling como herramientas de tokenizado y detección de oraciones en la etapa de pre procesamiento del conjunto de datos para utilizar NeuroNER.

Un detalle no menor es que todas las herramientas generaron errores al tokenizar las referencias a los distintos objetos legales, como es el caso de SpaCy y Stanford en la referencia *"...arts. 215 y conc. C.G.P..."* del ejemplo. Una consecuencia de estos errores y del formato de las referencias es que la etapa de pre procesamiento de los datos realizada por NeuroNER para transformar los mismos de formato BRAT a formato CoNLL genera entidades erróneas debido a que algunas referencias se dividían en dos o más dependiendo de cómo se había comportado la separación de oraciones. Por ejemplo la referencia a la sentencia: *"...SENTENCIA Nº 136 Montevideo, diez de junio de dos mil nueve. TRIBUNAL DE APELACIONES EN LO CIVIL*

DE CUARTO TURNO...” era divida en dos entidades, una que iba hasta *“...nueve.”* y otra que abarcaba hasta *“...TURNO...”*.

Para mitigar este problema, y evitar que la calidad del entrenamiento se vea afectada debido al conjunto de datos, se procedió a modificar el código fuente del sistema con el objetivo de que no separe las entidades anotadas en formato BRAT, si no que las tome como un elemento único aunque el tokenizador las haya separado generando dos o más entidades en oraciones diferentes.

5.1.5. Evaluación y Resultados

En esta sección se detallarán las distintas pruebas realizadas con NeuroNER, comparando los resultados obtenidos y analizando su comportamiento. Como se mencionó anteriormente, la utilización de esta herramienta se divide básicamente en dos enfoques, y estos dependen de la herramienta elegida para la detección de oraciones y su correspondiente tokenización, las cuales en este caso son SpaCy y Freeling.

Multiclase con todos los objetos legales

La primera prueba realizada consistió en entrenar el sistema con ejemplos de todos los objetos legales, no pudiendo obtener buenos resultados. Una de las razones de este comportamiento es el gran desbalance que existe en el corpus entre los distintos objetos. Como se puede ver en la figura 5.3, en el conjunto de entrenamiento hay anotadas más de 5000 referencias a artículos, mientras que menos de 1000 son referencias a decretos.

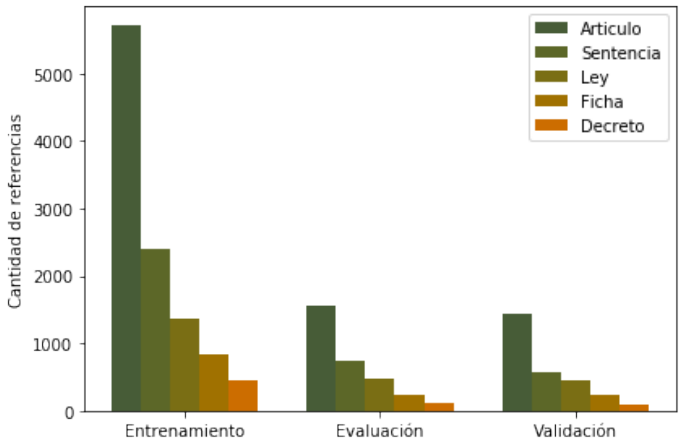

Figura 5.3: Cantidad de referencias por conjunto

Dentro de las diferentes alternativas para solucionar este problema se manejaron técnicas de balanceo consistentes en agregar o quitar referencias, pero finalmente se terminaron descartando. Un factor determinante que influyó en la decisión de no utilizar técnicas de balanceo fue que el conjunto de entrenamiento a utilizar estaba compuesto por textos de sentencias en los cuales, además de existir referencias anotadas a objetos legales, también existía texto no etiquetado necesario para componer el contexto de la sentencia.

Para quitar referencias es necesario disminuir la cantidad de ejemplos de las clases que poseen más, como ser las clases Artículo y Sentencia. Pero esto implicaría quitar del conjunto de entrenamiento algunas sentencias que posean muchas referencias a estos objetos legales o quitar oraciones dentro de las cuales ocurren estas referencias. En el primer caso, quitar sentencias enteras no garantiza que el conjunto de datos se balancee, ya que en la mayoría de estas hay ocurrencias de varios objetos legales. Por otro lado, el quitar oraciones conteniendo determinadas referencias también podría tener un impacto negativo en el entrenamiento, debido a que se estaría quitando contexto a la sentencia.

El caso de agregar referencias presenta algunos inconvenientes similares a los mencionados en el párrafo anterior, ya que para agregar más ocurrencias de referencias de un determinado objeto legal, por ejemplo Decreto, se tendrían que generar oraciones que contengan nuevas referencias lo cual volvería a generar problemas de contexto además de agregar complejidad al proceso.

Token	Sentencia	Inicio	Fin	Etiqueta	Anotación
art	1337	2308	2311	B-Articulo	Articulo
.	1337	2311	2312	I-Articulo	Articulo
37	1337	2313	2315	I-Articulo	Articulo
de	1337	2316	2318	I-Articulo	Articulo
la	1337	2319	2321	I-Articulo	Articulo
Ley	1337	2322	2325	I-Articulo	Articulo, Ley
No	1337	2326	2328	I-Articulo	Articulo, Ley
.	1337	2328	2329	I-Articulo	Articulo, Ley
17.243	1337	2330	2336	I-Articulo	Articulo, Ley

Tabla 5.2: Superposición de referencias

Otro gran problema que presenta el enfoque que incluye a todos los objetos legales, es el de las anotaciones superpuestas. Esto es un problema debido a que NeuroNER, al transformar los datos de formato BRAT a formato CoNLL, genera a partir de los mismos una lista de entidades anotadas indicando para cada una, el caracter de comienzo y el caracter de fin, para luego tokenizar la misma y asignarle a los tokens las etiquetas B o I según corresponda. Por lo tanto, es razonable que un token no pueda tener asignadas dos etiquetas, por lo que en dicho proceso se pierden anotaciones. El efecto que provoca esta pérdida se ve reflejado en el ruido que se introduce en los datos, debido a que como se ve en el ejemplo 5.2, el texto *"...art. 37 de la Ley No. 17.243..."* presenta una anotación de una referencia a artículo y una anotación de una referencia a ley, pero al pasar por el pre procesamiento de NeuroNER, se pasa a tener una única referencia, la primera que ocurre. Para entender un poco mejor la magnitud de este inconveniente se calculó la cantidad de referencias anotadas que fueron ignoradas por NeuroNER en el conjunto de entrenamiento. El resultado fue de 1501 referencias, lo cual corresponde al 13.9 % del total de referencias incluidas en el conjunto de datos mencionado.

	Precision	Recall	F1-Score
Articulo	79.15	76.52	77.81
Decreto	0.00	0.00	0.00
Ficha	89.41	87.19	88.28
Ley	42.20	32.62	36.80
Sentencia	65.16	68.61	66.84
Micro-Avg	74.42	71.20	72.77

Tabla 5.3: Resultados SpaCy

	Precision	Recall	F1-Score
Articulo	85.57	83.49	84.52
Decreto	0.00	0.00	0.00
Ficha	78.15	87.19	82.42
Ley	42.18	43.97	43.06
Sentencia	64.02	66.71	65.34
Micro-Avg	76.38	75.24	75.80

Tabla 5.4: Resultados Freeling

Las tablas 5.3 y 5.4 muestran los resultados de evaluar el modelo entrenado con NeuroNER utilizando SpaCy y Freeling respectivamente para la detección de oraciones y teniendo en cuenta todos los objetos legales. En los valores de las referencias a Decretos y Leyes se pueden ver las consecuencias de los problemas mencionados anteriormente. Debido a que la cantidad de ejemplos de referencias a los demás objetos legales es muy superior a la cantidad de referencias a los objetos mencionados, y que se introducen varias referencias con sus tokens etiquetados con "O" debido al problema de superposición, el modelo no logra aprender los patrones de estas confundiéndolas con referencias a otras entidades o con texto no correspondiente a una referencia. Además, aunque como se mencionó al final de la sección 5.1.4, se modificó el código para evitar que tanto Freeling como SpaCy dividieran las referencias, la herramienta que obtuvo mejores resultados fue Freeling. Este resultado se debe a que como muestra la figura 5.1, el conjunto de entrenamiento se procesa a nivel de oración siendo importante entonces el contexto en el cual se encuentran las referencias a los objetos legales y qué tan bien se determinaron los límites entre las oraciones que las incluyen. En este aspecto Freeling se comportó de mejor manera (sección 5.1.4) por lo que obtuvo mejores resultados en esta prueba.

Dadas estas observaciones se decidió que para obtener mejores resultados, en vez de entrenar un único modelo que reconozca las referencias a todos los objetos legales seleccionados como de interés, resultaba mejor entrenar varios modelos combinando aquellos objetos cuyas referencias no se superpusieran y cuya cantidad de ocurrencias en el conjunto de entrenamiento no fueran tan distantes. Debido a estas restricciones se entrenaron tres modelos, uno encargado de extraer referencias a Artículos (incluyendo el texto de su fuente), otro encargado de extraer referencias a Sentencias y Leyes y el último encargado de extraer referencias a Decretos y Fichas.

Otro punto importante a destacar es que debido a la leve pero existente diferencia entre los resultados que favorece la utilización de Freeling en el proceso de entrenamiento, se decidió realizar las pruebas posteriores teniendo en cuenta únicamente esta herramienta para las tareas de separación del texto en oraciones y tokenización.

Separando objetos legales

En el corpus utilizado, el objeto legal al que más referencias hay es el Artículo, presentando una diferencia considerable con el segundo objeto legal más referenciado (Sentencia) como se puede ver en la figura 5.3. Esto sumado a que presenta anotaciones superpuestas con los objetos legales Decreto y Ley, fue la razón por la cual se decidió entrenar un modelo individual para este tipo de referencias. Además, los formatos de

estas referencias son muy heterogéneos, transformando el problema en no sólo distinguir qué es una referencia a artículo y qué no, si no distinguir exactamente qué secuencia de tokens conforman la referencia completa.

El segundo modelo entrenado tomó en cuenta a las Sentencias y Leyes, debido a que no presentan anotaciones superpuestas entre sí, los formatos de sus referencias son notoriamente diferentes y dentro de los objetos legales restantes son las que presentan cantidades de ocurrencias más similares. Por último, y por las mismas razones recién mencionadas, se entrenó un modelo encargado de reconocer referencias a Fichas y Decretos.

	Precision	Recall	F1-Score
Artículo	88.75	87.84	88.30
Sentencia	77.65	70.79	74.06
Decreto	83.51	72.97	77.88
Ley	92.84	91.68	92.26
Ficha	95.24	85.27	88.40

Tabla 5.5: Resultados entrenando modelos separados

La tabla 5.5 muestra los resultados de evaluar los tres modelos contra el conjunto de evaluación. Si se tienen en cuenta los resultados obtenidos por el modelo entrenado con todos los objetos legales, los cuales se muestran en la tabla 5.4, se puede apreciar una notoria mejoría resultado de separar los tipos de referencias al entrenar. Los objetos legales para los cuales se obtuvieron mejores resultados fueron Ley, Ficha y Artículo. La figura 5.4 muestra que las referencias a los dos primeros en la mayoría de los casos no se las encuentra en formatos muy diferentes debido a que al citar una ley generalmente se da el número de la misma (ejemplo III) y en algunas ocasiones también el nombre por el cual se la conoce (ejemplo IV), mientras que las referencias a fichas en la mayoría de los casos están presentes en el formato IUE (Identificador Único de Expediente) como se puede ver en los ejemplos I) y II).

I) *IUE 0227-000149/2013*

II) *FICHA Nº 370-359/2003*

III) *Ley No. 19.090*

IV) *Ley Orgánica Policial*

Figura 5.4: Ejemplos formatos Ley y Ficha

El objeto legal para el cual se obtuvieron peores resultados fue la Sentencia. Una de las razones de este comportamiento es que las referencias a sentencias son las que se presentan en formatos más distintos, como se puede ver en 5.5. Además, los resultados de la tabla 5.5 surgen de la evaluación CoNLL mencionada en la sección 5.1.2, la cual compara entidades completas teniendo una probabilidad de error proporcional a la cantidad de tokens que conforman a la referencia. Por lo tanto, debido a que una sentencia se identifica por número, sede, turno y fecha, las referencias a las mismas están compuestas por una cantidad grande de tokens, aumentando la probabilidad de que el modelo no reconozca exactamente las secuencias que forman a las mismas.

I) *sentencia interlocutoria DFA-0014-000153/2014 SEI-0014-000026/2014 de 31 de marzo de 2014*

II) *DFA-0008-000072/2016 SEI-0008-000027/2016 TRIBUNAL DE APELACIONES EN LO CIVIL DE SEPTIMO TURNO*

III) *Resolución No. 1614/2015 dictada a fs. 158-165 por el Juzgado Letrado de Primera Instancia de Maldonado de 6º Turno*

Figura 5.5: Ejemplos formatos Sentencia

Con estas observaciones se quiere llegar a la conclusión de que un resultado de 74.06 % en medida F1 no es malo teniendo en cuenta las condiciones dadas. La tabla 5.6 muestra los resultados obtenidos para Sentencia pero teniendo en cuenta la evaluación BIO, la cual realiza una comparación token a token. Se puede ver que en medida F1 para las etiquetas B-Sentencia e I-Sentencia se obtuvieron un 86.00 % y 90.48 % respectivamente. Esto demuestra que los Falsos Negativos de la evaluación CoNLL son casos en los cuales el modelo etiquetó correctamente a la mayoría de los tokens que componen la referencia.

	Precision	Recall	F1-Score
B-Sentencia	90.16	82.20	86.00
I-Sentencia	90.08	90.87	90.48

Tabla 5.6: Resultados Sentencia evaluación BIO

Luego de detallar las diferentes pruebas realizadas, es necesario hacer referencia al criterio de parada utilizado en el entrenamiento. Este criterio es el configurado por defecto en NeuroNER y consiste en culminar el entrenamiento cuando no se detecta una mejora en la medida F1 sobre el conjunto de validación durante más de 10 iteraciones seguidas. Otro punto a destacar, es la utilización de los parámetros del sistema. Los resultados finales surgen de entrenamientos utilizando los parámetros recomendados por [21], ya que dicha configuración es la que aporta mejores resultados sobre la mayoría de las áreas de trabajo donde se utiliza NER. También es importante mencionar que los entrenamientos se realizaron sobre una máquina con las siguientes características:

- **Procesador:** Intel Core i7-4510U 2.00GHz
- **Memoria RAM:** 8GB
- **Disco Duro:** 1TB

En cuanto a los tiempos de entrenamiento, la evaluación sobre todos los objetos legales conllevó poco más de 21 horas, mientras que la suma de los entrenamientos mencionados en 5.1.5 superó las 35 horas. Con el objetivo de disminuir estos tiempos, se experimentó con el uso de GPU para el entrenamiento utilizando una instancia de Amazon Web Services [28], no obteniendo mejorías notorias. Esto coincide con lo indicado en un hilo del foro GitHub [29] del proyecto NeuroNER, donde uno de los creadores menciona que el uso de GPU no mejora notoriamente el rendimiento.

5.2. Campos Aleatorios Condicionales (CRF)

Luego de realizadas las pruebas con el sistema NeuroNER, y dados los bajos resultados obtenidos al entrenar el modelo con ejemplos de todos los objetos legales surgió la incógnita de qué atributos estaba tomando la red neuronal. Esta duda se presentó debido a que suena lógico que si por ejemplo, en un fragmento de texto aparece la palabra "decreto" o "ley" seguida por un número, seguramente se trate de una referencia a dichos objetos legales. Es por esta razón que se decidió experimentar con un método de aprendizaje automático en el cual los atributos fueran explícitos.

Un método que requiere ingeniería de atributos y es conocido por su capacidad de aplicación en problemas de etiquetado de secuencias es el conocido como "Campos Aleatorios Condicionales". Una de las principales características de este método es que el contexto en el cual se encuentra un token, es tenido en cuenta para la correcta clasificación del mismo. Además al ser el usuario quien define los atributos a utilizar, presenta más flexibilidad para poder evitar caer en asunciones erróneas las cuales pueden ocurrir en el caso de las redes neuronales.

5.2.1. Utilización

Para este trabajo se utilizó la implementación de CRF disponible en la herramienta CRFsuite de sklearn [30]. El primer paso consistió en la selección de atributos, la cual se encuentra detallada en la sección 5.2.2. Luego, se procedió a calcular el valor de los atributos para todos los tokens de todas las sentencias que componen el conjunto de entrenamiento. El formato de los datos de entrada del algoritmo consiste en dos niveles de listas. En la lista padre cada elemento se corresponde con una sentencia de la BJN. A su vez, cada sentencia de la BJN es representada por una lista de triplas donde cada una de ellas representa a un token existente en el texto de la sentencia. La información contenida en cada una de las triplas que representan tokens está dada por: el token en texto plano, un diccionario con los valores de los atributos para dicho token y la etiqueta real del token en formato BIO. En este proceso se incluyó el uso de la herramienta Freeling tanto para reconocer oraciones y tokenizarlas como para obtener las categorías gramaticales (POS-tag) tanto del token en cuestión como de sus vecinos.

5.2.2. Atributos seleccionados

En cuanto a la selección de atributos requeridos para entrenar un CRF, se tomaron en cuenta aquellos que brindan información acerca del token, su composición y su contexto. Es por esto que se siguió las recomendaciones indicadas en Konkol [12], donde se sugiere utilizar atributos ortográficos, sufijos y prefijos, categorías gramaticales y lemas entre otros. Por lo tanto, como atributos individuales del token se utilizaron utilizaron los presentados en la tabla 5.7 para el token "Sentencia". Además, el autor sugiere la definición de una ventana para extraer información sobre el contexto del token. En este caso se consideró una ventana de tamaño 1 la cual abarca al token inmediatamente anterior y siguiente al considerado (siempre y cuando no se encuentre al comienzo o final de una sentencia). En ambos casos, se tomaron en cuenta los atributos detallados en la tabla 5.8, en este caso para los tokens "la" y "número".

El token en minúscula	**Sentencia -> sentencia**
Los últimos 3 elementos del token	**Sentencia -> cia**
Los últimos 2 elementos del token	**Sentencia -> ia**
Si el token esta completamente en mayúscula	**Sentencia -> False**
Si su primer caracter esta en mayúscula y los demás están en minúscula	**Sentencia -> True**
Si el token es un dígito	**Sentencia -> False**
El lema del token	**Sentencia -> sentencia**
La etiqueta gramatical del token	**Sentencia -> NCFS000**
Los primeros 2 caracteres de la etiqueta gramatical	**Sentencia -> NC**

Tabla 5.7: Atributos del token

El token en minúscula	**la -> la**	**número -> número**
Si el token esta completamente en mayúscula	**la -> False**	**número -> False**
Si su primer caracter es mayúscula	**la -> False**	**número -> False**
La etiqueta gramatical del token	**la -> DA0FS0**	**número -> NCMS000**
Los primeros 2 caracteres de la etiqueta gramatical	**la -> DA**	**número -> NC**

Tabla 5.8: Atributos del token anterior y siguiente

5.2.3. Evaluación y Resultados

Los resultados presentados a continuación fueron obtenidos luego de realizar una optimización de hiperparámetros, más específicamente de los parámetros de regularización debido a que ésta es utilizada para que el CRF no se ajuste completamente a los datos de entrenamiento y así pueda generalizar en datos aún no conocidos. Para esto, siguiendo las recomendaciones de la documentación de CRFsuite [31], se utilizó como algoritmo de optimización a L-BFGS el cual utiliza una regularización Elastic Net combinando las penalizaciones L1 y L2. Para obtener los mejores coeficientes se utilizó el método RandomizedSearchCV de sklearn mediante una validación cruzada de tamaño 5 y tomando como métrica de comparación a la medida F1. Los coeficientes de regularización obtenidos fueron los siguientes:

- L1 = 0.5791
- L2 = 0.0349

Como se puede ver en la tabla 5.9, los resultados son bastante similares a NeuroNER luego de entrenar los objetos legales por separado, pero superior al entrenamiento de todos los objetos legales juntos, confirmando la hipótesis sobre el peso del token con el nombre del objeto legal mencionado anteriormente.

	Precision	Recall	F1-Score
Artículo	89.42	86.38	87.87
Sentencia	78.96	74.76	76.80
Decreto	81.66	74.24	77.77
Ley	85.03	88.65	86.80
Ficha	98.70	94.60	96.61
Micro-Avg	86.82	83.96	85.36

Tabla 5.9: Resultados CRF

Además, se observó que, como se mencionó en la sección 5.1.1, CRF aprende determinadas restricciones sobre el orden de las etiquetas. A continuación se pueden ver las transiciones de etiquetas más y menos probables aprendidas por el modelo CRF, según el peso asignado por el modelo a cada una.

From \ To	O	B-Articulo	I-Articulo	B-Decreto	I-Decreto	B-Ficha	I-Ficha	B-Ley	I-Ley	B-Sentencia	I-Sentencia
O	4.299	0.724	-8.99	-0.107	-5.896	0.141	-5.181	-0.202	-5.113	-0.085	-7.867
B-Articulo	-4.838	-3.141	2.988	0.0	-2.345	-0.623	-1.479	-0.287	-1.731	-1.165	-2.901
I-Articulo	-0.707	-1.592	4.398	-2.021	-3.904	-2.251	-2.64	-1.749	-3.233	-0.94	-3.325
B-Decreto	-3.298	-0.607	-3.123	0.0	2.958	-0.317	-1.509	0.0	-1.003	-1.092	-2.11
I-Decreto	-0.614	-0.743	-3.079	0.011	5.049	-0.033	-0.931	-0.009	-1.227	-0.601	-2.18
B-Ficha	-2.339	-0.762	-2.224	0.0	-1.841	-1.327	3.833	0.0	-0.885	-0.976	-2.299
I-Ficha	-0.716	-0.452	-2.474	0.0	-1.084	-1.546	4.78	-0.017	-0.691	-0.693	-2.571
B-Ley	-2.342	-0.079	-4.581	0.0	-1.504	0.0	-0.763	0.0	4.229	0.0	-1.354
I-Ley	-0.983	-0.726	-3.37	-0.087	-1.477	0.0	-0.679	-0.27	5.236	-0.123	-1.436
B-Sentencia	-4.342	-1.215	-3.669	-0.225	-2.848	-2.642	-2.409	-0.138	-1.893	-1.937	3.192
I-Sentencia	-0.766	-1.462	-3.943	-0.817	-3.051	-2.331	-2.475	-0.829	-2.523	-2.684	5.217

Figura 5.6: Transiciones CRF

En esta imagen se puede ver que las transiciones más probables corresponden correctamente a aquellas en las que se pasa de una etiqueta B a una etiqueta I de la misma clase. A su vez, también se ve claramente que el modelo toma como negativo el pasar de una etiqueta O a una etiqueta I (lo cual es imposible en la práctica), como se puede ver en el caso $O \Rightarrow I - Articulo$ el cual tiene el puntaje más bajo.

y=B-Ley top features		**y=I-Ley** top features	
Weight[?]	**Feature**	**Weight**[?]	**Feature**
+2.536	-1:word.lower():19120	+2.481	-1:word.lower():ley_nº
+2.088	word.lower():ley_nº	+2.392	-1:word.lower():ley_no
+2.088	word.lemma():ley_nº	+2.066	-1:word.lower():8.733
+1.968	word.lemma():ley_nro	+1.884	word.lower():19090
+1.968	word.lower():ley_nro	+1.712	-1:word.lower():ley_nro
+1.932	word.lower():ley_no	+1.578	-1:word.lower():ley
+1.932	word.lemma():ley_no	+1.288	+1:word.lower():el_tribunal_resuelve
+1.903	+1:word.lower():19090	+1.258	word.lemma():18572
+1.389	word.lower():en_la_redaccion_dada_por_la_ley_no	+1.209	word[-3:]:572
	… 126 more positive …		*… 519 more positive …*
	… 37 more negative …		*… 84 more negative …*
-1.718	bias	-1.470	+1:word.istitle()

Figura 5.7: Atributos principales consideradas para Ley

En la figura 5.7 se pueden ver los principales atributos que el modelo CRF tomó en cuenta para clasificar tokens como B-Ley y como I-Ley. Con esta información se puede reafirmar lo dicho anteriormente sobre que las ocurrencias de palabras que identifican a los objetos legales (por ejemplo: "ley", "articulo", "art."), deben ser lo suficientemente importantes a la hora de determinar si un segmento de texto se corresponde con una referencia o no. Esto se ve reflejado en la lista de atributos con más peso en el CRF a la hora de decidir si es o no el comienzo de una referencia a una ley, ya que la mayoría de estos implican que el token correspondiente a la etiqueta B-Ley debe incluir a la palabra "ley". Otro atributo considerado importante en este caso es que el token tenga como sucesor un número con 5 dígitos, como es el caso de la "ley 19090". De igual manera, para el caso de decidir si un token se encuentra dentro de una ley, se puede observar que la mayor parte del peso de esta decisión está relacionada con que la palabra antecesora contenga el token "ley", seguido de que sea un número de 5 dígitos. En este último caso se puede ver también que el hecho de que la palabra sucesora comience en mayúscula, expresado como "+1:word.istitle()", no es de importancia a la hora de tomar la decisión.

5.3. SpaCy

Otra alternativa utilizada a la hora de extraer las referencias se basó en la utilización del NER proporcionado por la herramienta SpaCy. Esta decisión se tomó ya que este sistema emplea CNN (Redes Neuronales Convolucionales), las cuales están siendo utilizadas cada vez más en el ámbito del procesamiento de lenguaje natural y mejorando los resultados en muchas áreas del mismo, pudiendo dar lugar a mejores resultados que NeuroNER. Más precisamente, se utilizan redes neuronales convolucionales profundas, con conexiones residuales. Las redes neuronales con conexiones residuales son aquellas que poseen conexiones entre capas no contiguas, con el objetivo de evitar el problema de la desaparición de gradientes, el cual provoca que el gradiente se estanque y la red no pueda seguir entrenándose. Esto se logra reutilizando las activaciones de una capa anterior hasta que la capa adyacente aprenda sus pesos.

SpaCy es una librería de código abierto utilizada para el procesamiento de lenguaje natural, escrito en Python y Cython. Está conformada por varias herramientas dentro de las que se destacan el tokenizador, separador de oraciones utilizado en la sección 5.1.4, parser de dependencias, reconocimiento de entidades nombradas, entre otros. Además, para varias de estas herramientas (principalmente el tokenizador) presenta soporte en diferentes idiomas como por ejemplo inglés, español y alemán. Una ventaja de esta herramienta es que permite entrenar y utilizar modelos estadísticos generados por librerías como TensorFlow, Keras, Scikit Learn o PyThorch, sin tener que implementarlos.

De acuerdo a la web oficial de SpaCy, la arquitectura actual del sistema NER no ha sido publicada. De igual manera, se conoce que la arquitectura general del sistema SpaCy esta formada por un pipeline de herramientas, entre las cuales se encuentra el NER. Esto permite construir pipelines de procesamiento de datos personalizados, ya sea deshabilitando algunas herramientas o personalizándolas. En este trabajo, es de interés utilizar solamente el NER, por lo tanto se procedió a deshabilitar los demás componentes del pipeline y entrenar un modelo “en blanco” para el Español, con las entidades nombradas legales anotadas previamente. Un modelo “en blanco” significa que el mismo no posee ningún entrenamiento previo, ya que generalmente al utilizar esta herramienta, se cargan modelos pre-entrenados para determinadas categorías como por ejemplo: Personas, Nombres y Organizaciones. De esta forma, el único componente de la herramienta que procesará las entradas será el NER, el cual se centrará en reconocer solamente objetos legales.

5.3.1. Utilización

Para el entrenamiento del NER, fue necesario modificar el formato de los datos ya que SpaCy no admite que los mismos tengan la estructura de salida de BRAT. Es por esto que se modificaron las anotaciones obtenidas en formato BRAT, para que sean compatibles con el formato admitido por SpaCy.

Como se puede ver en 5.8, el formato de entrada de SpaCy consiste en una cadena de texto (por lo general oraciones), seguido de una lista con las entidades dentro de la misma, en la cual cada elemento es una tupla con los índices de comienzo y fin de la entidad en el texto y el tipo de entidad a la cual corresponde. Como se comentó anteriormente, la separación en oraciones no fue muy satisfactoria con ninguna herramienta, por lo que se decidió en vez de realizar el procesamiento por oraciones, hacerlo por sentencia. Es decir, que las cadenas de texto de cada objeto de entrada se correspondían al texto de una sentencia.

"...Por lo tanto, si los bienes eran gananciales, se debía amparar la reivindicación del 50 % que se había pretendido contra AA, a cuyo respecto no cabía extender lo decidido por ***sentencia Nº 179/2007 del T.A.C. 1º*** *que se había pronunciado ... El Tribunal, con el voto coincidente de sus miembros (****art. 61 de la Ley Nº 15.750****), acordó confirmar la sentencia apelada, ... se los condenará a pagar el 50 % del valor venal de los inmuebles de autos que se determinará por el procedimiento previsto por el* ***art. 378.1 del C.G.P.*** *y con el límite de lo solicitado al demandar (U$S 40.000), todo ello por los fundamentos que se pasan a exponer..."*

```
'entities': [(1566, 1601, 'Sentencia'), (2919, 2946, 'Articulo'), (3321, 3342, 'Articulo')]
```

Figura 5.8: Formato de entrada SpaCy

Para el entrenamiento, como se comentó anteriormente, se procedió a crear un nuevo modelo "en blanco" para el idioma español, y deshabilitar los demás componentes del pipeline, de manera de entrenar solamente el NER. El procedimiento consistió en realizar diferentes entrenamientos variando el dropout como técnica de regularización y evaluando los resultados de cada iteración sobre el conjunto de validación utilizado en la sección 5.1. En cuanto a los criterios de parada, además de un máximo de 100 iteraciones se utilizó el mismo criterio que en NeuroNER, el cual consistía en cortar el entrenamiento si en 10 iteraciones consecutivas no se mejoraba el valor de la medida F1 sobre el conjunto de validación.

5.3.2. Evaluación y Resultados

En la figura 5.9 se pueden observar los valores de la medida F1 a medida que avanzaba el entrenamiento para cada valor de dropout. Se puede ver que a medida que aumentaba el dropout, la convergencia se volvía más lenta, mientras que a menor dropout el modelo convergía rápidamente culminando el entrenamiento debido al criterio de parada aplicado sobre la medida F1, como fue el caso de dropout 0.1 donde el entrenamiento culminó a las 25 iteraciones. Finalmente, el mejor valor para la medida F1 sobre el conjunto de validación fue de 90.69 % y se alcanzó en la iteración 33 con un valor de dropout de 0.35. Este modelo fue el elegido a la hora de calcular las diferentes métricas sobre el conjunto de evaluación.

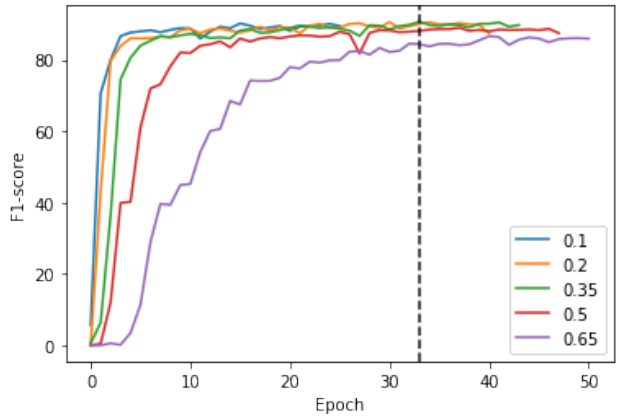

Figura 5.9: Medida F en función de epochs en validación

Los resultados obtenidos entrenando el NER de SpaCy de la forma mencionada anteriormente fueron notoriamente superiores en la medida F1 en comparación con NeuroNER, tanto de manera general como en cada objeto legal por separado, así como en comparación con el sistema CRF. Como se puede ver, entrenando todos los objetos legales juntos con un dropout de 0.35 se obtiene una medida F1 de más de 87 % en total, y se obtienen valores diferentes de 0 para Decreto y claramente mayores para Ley.

	Precision	Recall	F1-Score
Artículo	93.05	91.18	92.10
Sentencia	78.11	76.60	77.35
Decreto	87.80	92.30	90.00
Ley	87.95	94.80	91.25
Ficha	85.14	95.51	90.03
Micro-Avg	87.63	87.63	87.63

Tabla 5.10: Resultados SpaCy

Estos resultados, al igual que en los métodos anteriores, se obtuvieron de realizar una evaluación a entidad completa, en este caso utilizando la librería Scorer de SpaCy. Es decir, que para que un ejemplo sea tomado como correcto, la tripla (caracter de inicio, caracter de fin y entidad) predicha debe coincidir totalmente con la entidad anotada.

Una aclaración pertinente a realizar es que en todos los casos, al momento de evaluar el rendimiento de los modelos sobre el conjunto completo de clases, se tomó como métrica a la medida micro. Esta métrica penaliza de menor forma cuando hay desbalance de clases ya que toma la suma de los valores de cada clase para construir cada término. Es decir, los términos Verdadero Positivo, Falso Negativo y Falso Positivo utilizados para calcular una medida micro, surgen de sumar los correspondientes valores obtenidos para cada clase. Por otro lado, si se hubiera tomado como métrica a la medida macro, la cual realiza un promedio entre las medidas obtenidas individualmente, se hubiera favorecido aún más a SpaCy en comparación con los otros modelos entrenados. Esto se debe a que al tomar todas las clases con igual peso, un valor bajo para una clase con poco peso en cuanto a cantidad de ejemplos, afecta más al resultado total que si se tomara una medida micro. Un ejemplo de esto se puede ver entre el modelo CRF y SpaCy, ya que en el primero se obtuvo un 77.77 % de F1 y en el segundo un 90.00 % para la clase Decreto, lo cual finalmente resulta en un 85.17 % de medida macro-F1 para CRF y 88.15 % de medida macro-F1 para SpaCy.

Capítulo 6

Otros procesos sobre la BJN

En esta sección se detallan aspectos relacionados a otras tareas realizadas sobre la BJN para cumplir con el alcance inicial del proyecto. Más precisamente, se tratan dos temas, la extracción de referencias a Doctrinos, la cual se realizó con métodos diferentes a los detallados en el capítulo anterior, y la clasificación semántica de las referencias, etapa finalmente no desarrollada debido a razones detalladas en la sección 6.2.

6.1. Extracción de Doctrinos

Como se mencionó al comienzo del capítulo 4, a pesar de que las referencias a Doctrinos se consideraron relevantes en el marco de este trabajo, su reconocimiento y extracción se postergó debido a distintas razones como ser la falta de ejemplos etiquetados y la falta de una lista inicial de Doctrinos conocidos. Debido a la complejidad de esta tarea, y teniendo en cuenta lo mencionado en Konkol [12], donde se indica que un método simple para la tarea de NER basada en reglas es contar con un diccionario de entidades conocidas, se decidió generar una lista primaria de Doctrinos a partir de las sentencias de la BJN. Para lograr este cometido, se procedió a hacer uso de uno de los sistemas generados en Rosá [4] para obtener una lista de candidatos a Doctrinos, sobre la que luego se realizó un estudio para acotarla a Doctrinos válidos en el ámbito legal. Resulta de interés obtener este lista para poder realizar análisis posteriores sobre la influencia que tiene cada uno sobre determinadas áreas legales. En esta sección se presentan detalles sobre el sistema utilizado y el proceso seguido para cumplir con los objetivos planteados.

6.1.1. Identificación de opiniones de diferentes fuentes en textos en Español

El trabajo referenciado anteriormente consiste en una investigación acerca de las distintas formas de transmitir opiniones en diferentes textos y los diferentes conceptos que están relacionados en una opinión, como ser el asunto, el mensaje y la fuente. Este último refiere a la entidad que transmite dicha opinión y resulta de interés en este trabajo debido a que las referencias a Doctrinos muchas veces forman parte de una argumentación o de una opinión expresada sobre algún tema en particular, como se puede ver en los ejemplos de la figura 6.1.

I) *El **Dr. Gastón Chaves** en "Estudios sobre el Nuevo Proceso Penal" **expresó** que en la nueva normativa "...resulta intacta la facultad del Juez de ponderar si la cuantía del daño está en razonable correspondencia con el propio fundamento del castigo, que es precisamente la protección del bien dañado"*

II) *El **Prof. Plá Rodríguez señala que**: '...Es el trabajador el que debe producir la prueba pero ella debe ser examinada objetiva e imparcialmente, teniendo en cuenta las dificultades existentes para la obtención de esa prueba y la obligación de los empleadores de suministrar los elementos documentales que tienen a su disposición y que servirán para esclarecer el punto.'*

III) *Como **explica Giorgi**, en concepto común a los procesos de anulación de actos administrativos y de inconstitucionalidad de la Ley, interés directo significa un interés inmediato, no eventual ni futuro.*

IV) Según ***Américo Plá Rodríguez**, el contrato de trabajo es aquel por el cual una persona se obliga a prestar una actividad en provecho y bajo la dirección de otra y esta a retribuirla.*

Figura 6.1: Ejemplos referencias a Doctrinos

En "Identificación de opiniones de diferentes fuentes en textos en Español" se desarrollaron tres sistemas diferentes para el reconocimiento de los distintos componentes de una opinión. El primero de ellos está basado en reglas contextuales y se enfoca en reconocer los siguientes componentes: predicado, asunto, fuente, mensaje y la opinión completa; el segundo sistema consiste en un modelo CRF enfocado exclusivamente en reconocer las fuentes de las opiniones, mientras que el tercer sistema se generó combinando los dos primeros también para el reconocimiento de fuentes de la opinión. Debido a que para el marco del presente trabajo no se contaba con los sistemas basados en CRF, se utilizó únicamente el sistema basado en reglas.

Este sistema contiene un total de 173 reglas agrupadas en los módulos de predicado, asunto, mensaje y fuente junto a otros módulos auxiliares utilizados para la conformación de la opinión completa. Estas reglas a su vez consultan diferentes componentes externos como un repertorio de predicados de opinión desarrollado exclusivamente para dicho trabajo, así como una lista de indicadores de persona (señor, doctor, candidato), una lista de verbos soporte (realizar, emitir) y una lista de términos que introducen a un asunto (sobre, en cuanto a, en lo que respecta a). Todo esto es necesario para el reconocimiento de los componentes mencionados anteriormente, dentro de los cuales se encuentra la fuente que es el que resulta de interés para la extracción de referencias a Doctrinos.

Para su utilización se requiere que los textos de entrada sean pre-procesados por la herramienta de POS-tagger de Freeling. Luego, el sistema de reglas se aplica secuencialmente módulo por módulo generando una salida en formato XML. Todas las reglas de un mismo módulo se ejecutan en una recorrida del texto de entrada y generan etiquetas en el texto cada vez que encuentran una coincidencia. Por ejemplo, cuando una regla correspondiente al componente "fuente" es exitosa, se etiqueta con dicha categoría al segmento de texto reconocido.

Como se mencionó recientemente la categoría y por ende el módulo de interés para este trabajo es el correspondiente a las fuentes. Algunos ejemplos de reglas utilizadas en dicho módulo son las que se ven en la figura 6.2, donde algunas de ellas cumplen el formato de las referencias a Doctrinos listadas como ejemplos anteriormente.

`f1a# [fuente] => no(prep) >> gn << (zona,3), vOpFinSF`	X dijo / X también opinó
`f1b# [fuente] => punt, vOpFinSF, (zona,3) >> gn <<`	.., dijo ayer X
`f1c# [fuente] => punt, vOpFinSF, (zona,3), prep, gn >> gn <<`	.., dijo a Brecha X
`f2# [fuente] => vOpPartSF, "por" >> gn <<`	expresado por X
`f3a# [fuente] => nOp, "de" >> gn <<`	apoyo de X
`f3b# [fuente] => >> gn << vSopFin, op(det), nOp`	X emitió una opinión
`f3c# [fuente] => nOp, vSopPart, "por" >> gn <<`	apoyo brindado por X
`f4a# [fuente] => "según", op(vOp) >> gn <<`	según X, ... / según dijo X, ...
`f4b# [fuente] => finOr, "para" >> gn <<`	. Para X, ...
`f4c# [fuente] => introdAut >> gn <<`	a juicio de X, ...
`f5a# [fuente] => "a" >> gn << (zona,3), vOpFinSA`	a X le molesta ...
`f5b# [fuente] => vOpFinSA, "a" >> gn <<`	... agrada a X

Figura 6.2: Ejemplos de reglas simplificadas para fuente. Extraído de [4]

6.1.2. Obtención de candidatos a Doctrinos

El proceso de extracción de referencias a Doctrinos comenzó entonces aplicando el sistema introducido en la sección anterior. Para esto se utilizaron las mismas 1000 sentencias elegidas aleatoriamente para lo realizado en las secciones 5.1, 5.2 y 5.3. Antes de aplicar las reglas contextuales fue necesario un paso de pre-procesamiento del texto de las sentencias seleccionadas. Este consistió en transformar las 1000 sentencias en un único archivo de texto, para luego aplicarle la función Analyzer de Freeling, la cual se encargó de etiquetar a cada token del texto con su respectiva etiqueta gramatical. Una vez realizado esto, y debido al gran consumo de tiempo que implicaba aplicar el sistema completo de reglas a todo el texto, se procedió a eliminar el módulo de reglas correspondiente al componente de la opinión "asunto". Este módulo era ejecutado luego del módulo de fuentes, por lo que no afectaba en lo absoluto la detección de las mismas, y ahorraba una cantidad de tiempo considerable.

El resultado fue un archivo XML conteniendo el texto de entrada etiquetado con las categorías restantes. Para obtener únicamente el texto entre las etiquetas <fuente>< /fuente> se utilizó la librería Beautiful Soup de Python, dedicada a facilitar la extracción de información desde archivos con formato HTML o XML. Luego de aplicar dicha librería se obtuvo una lista inicial de fuentes con una cantidad de elementos de 7731. Era de esperar que una gran cantidad de estos no se correspondieran con Doctrinos debido a que el sistema fue implementado para reconocer fuentes en el marco de opiniones, las cuales pueden ser de cualquier entidad. Algunos ejemplos de fuentes reconocidas que no se corresponden con Doctrinos son las siguientes:

I) *El Tribunal*

II) *los integrantes de la Sala*

III) *La Señora Juez de Primera Instancia*

IV) *el trabajador*

Figura 6.3: Ejemplos de fuentes reconocidas

6.1.3. Post Procesamiento de fuentes

Por las razones recién mencionadas fue necesario realizar un post-procesamiento para descartar aquellas fuentes que no correspondían a Doctrinos. Cabe destacar la dificultad de llegar a una lista con únicamente Doctrinos, debido a que es difícil distinguir entre uno de estos y un nombre propio cualquiera. La depuración de la lista se realizó de manera iterativa teniendo en cuenta diferentes propiedades y aspectos de las referencias a los Doctrinos.

Como primer paso, se hizo énfasis en las etiquetas gramaticales de los tokens que componían las fuentes. El ejemplo IV) de 6.3 está conformado por los tokens "el" y "trabajador", los cuales según Freeling se corresponden a un determinante, y un nombre común respectivamente. Este ejemplo claramente no se corresponde con un Doctrino, por lo tanto, una primera restricción realizada sobre la lista es que se consideraron fuentes válidas a aquellas que contuvieran al menos un token cuya categoría gramatical fuese nombre propio. De esta forma, mediante diferentes reglas y la aplicación de Freeling se pudo reducir la cantidad de candidatos a fuentes a 1943. Además de la restricción mencionada anteriormente, se impusieron otras más específicas debido a que algunos ejemplos contenían tokens etiquetados como nombres propios pero aún así no hacían referencia a una persona. Un ejemplo de esto se puede ver en el texto "el Juez interviniente", segmento cuya secuencia de categorías gramaticales es determinante-nombre propio-adjetivo. Para estos casos en particular se planteó la restricción de que la fuente debe culminar con un nombre propio.

Otra restricción que podría surgir a partir del ejemplo anterior sería la de descartar aquellas fuentes que contienen un adjetivo. Al realizar la prueba de esta regla, se pudo ver que se estaban ignorando fuentes que correspondían a Doctrinos, como por ejemplo "el ilustre administrativista Dr. Cajarville Peluffo" o "el profesor argentino Rafael Manóvil". Luego, otra regla utilizada fue descartar aquellas fuentes en las cuales existiera una preposición antes del nombre propio. Este caso se puede ver en la fuente "las representantes del Poder Legislativo", donde el token "del" es separado en "de" y "el" por Freeling, asignándole la categoría preposición. De esta manera se pudieron descartar muchos ejemplos que de otra forma hubieran sido difíciles de diferenciar de las fuentes válidas. Algunos ejemplos descartados por esta regla incluían términos correspondientes a relaciones entre personas, por ejemplo en muchos casos se repetían patrones como "el hermano de...", "los hijos de..." o "el socio de...", los cuales eran seguidos por nombres propios correspondientes a testigos o participantes del proceso judicial. Otro aspecto importante que se tuvo en cuenta en este proceso fue el hecho de que en muchas de las ocasiones en las cuales se hace referencia a un Doctrino, se utilizan calificativos como "Profesor" o Doctor". Si bien esto no ayudó a descartar fuentes no válidas, permitió confirmar a algunas como válidas, por ejemplo "Profesor Dr. Américo Plá Rodríguez".

Luego de la aplicación de este proceso iterativo para depurar la lista de posibles Doctrinos, se logró una lista con 374 elementos. Aún así, no había garantía de que la totalidad de los nombres obtenidos se correspondieran con Doctrinos, por lo que decidió realizar un último procesamiento de los datos para poder decidir cuales eran elegidos como válidos y cuales no. Para este último paso se tomó en cuenta la estructura de las sentencias de la BJN. En la mayoría de los casos, los textos de las sentencias se dividen en secciones, las cuales son: VISTO, CONSIDERANDO, RESULTANDO, y FALLA o RESUELVE. Si bien no todas las sentencias presentan explícitamente estas secciones, la mayoría cuenta con las más trascendentales como ser RESULTANDO y CONSIDERANDO. En la primera generalmente se realiza un repaso de lo resuelto en la sentencia de primera instancia así como un repaso de los hechos y declaraciones de los testigos. Por otro lado, en la sección CONSIDERANDO están presentes los argumentos sobre los cuales se basó el Tribunal para llegar a la resolución final. Es en esta sección donde usualmente aparecen las referencias a Doctrinos con el objetivo de argumentación o esclarecer determinados conceptos.

Por lo tanto, se decidió realizar una búsqueda de los nombres de la lista de fuentes en las 1000 sentencias utilizando el módulo `re` de la misma forma que en la sección 4.1.1. Además, se realizó una búsqueda de los nombres de las secciones antes mencionadas. Este proceso se aplicó con el objetivo de poder identificar en qué secciones del texto de una sentencia se encontraban las ocurrencias de los candidatos a Doctrinos presentes en la lista de fuentes. Esto fue posible debido a que el método `finditer` retorna las coincidencias y el rango de caracteres entre los que ésta aparece. Luego, se tomaron como referencia los caracteres entre los cuales se encontraba la sección CONSIDERANDO (calculados en base a las posiciones de las otras secciones) para clasificar la ocurrencia de una fuente con la etiqueta “IN” si aparecía dentro de la sección u “OUT” en los casos en los que el nombre aparecía en otra sección.

	Inicio	**Fin**	**Etiqueta**
López	171	176	OUT
VISTO	236	241	
RESULTANDO	648	658	
CONSIDERANDO	2346	2358	
ODRIOZOLA	3881	3890	IN
VÉSCOVI	4584	4591	IN
Plá Rodríguez	6323	6336	IN
RESUELVE	8023	8031	

Tabla 6.1: Post-procesamiento de fuentes

En la tabla 6.1 se pueden ver los resultados de aplicar lo antes mencionado a una determinada sentencia. Tres de los cuatro nombres reconocidos se encuentran dentro de la sección CONSIDERANDO, la cual abarca desde el caracter 2358 hasta el 8023. Si se extraen los segmentos de texto en los cuales estos nombres aparecen, se puede confirmar que éstos se corresponden con Doctrinos. En cambio, el fragmento de texto en el cual aparece el nombre “López” se encuentra al principio de la sentencia y corresponde con el apellido de una Ministra Firmante presente en la sala. Si bien en este caso no sucede, existen casos en los cuales una ocurrencia de una fuente que no es un Doctrino aparece dentro de la sección CONSIDERANDO o una ocu-

rrencia de un Doctrino aparece afuera. Para disminuir el error al descartar posibles Doctrinos, se aplicó una última instancia de filtrado teniendo en cuanta distintos factores como la cantidad de ocurrencias dentro de la sección CONSIDERANDO (etiquetadas como "IN"), la cantidad de ocurrencias fuera de dicha sección (etiquetadas como "OUT") y la cantidad de sentencias distintas en las cuales ocurre la fuente. Teniendo en cuenta que es de esperar que un Doctrino sea referido en más de una sentencia, y que difícilmente una fuente que no se corresponda a un Doctrino sea referenciada únicamente en la sección CONSIDERANDO, se llegó a la siguiente condición para determinar qué es un Doctrino y qué no:

$$f\texttt{:fuente} \in \texttt{Doctrinos} \Leftrightarrow p(f) > 0{,}5 \ \wedge \ c(f) > 1$$

donde,

$p(f) = \frac{|IN|}{|IN|+|OUT|}$

$c(f) =$ cantidad de sentencias donde ocurre f

Figura 6.4: Criterio para determinar Doctrino

De esta forma, los casos excepcionales mencionados anteriormente no afectan a la decisión final, debido a que suena razonable que el nombre de un Doctrino aparezca más veces dentro de la sección CONSIDERANDO que fuera de la misma.

I) *"...que, conforme al estado del proceso y la naturaleza de la cuestión planteada, la controversia debe resolverse teniendo en cuenta los términos de la demanda, (**ODRIOZOLA**, Judicatura Nº 10 pags. 245 y ss...")*

II) *"...en tanto el objeto del proceso se delimita por la pretensión allí contenida, que no será modificado por la contestación, salvo la eventual introducción de reconvención (Cfm. **VESCOVI**, Derecho Procesal Civil, T. II pags.64/69; 104/118; R.U.D.P. 2009/4 c. 37))..."*

III) *"...Como ha expresado **Plá Rodríguez**, "en caso de discordancia entre lo que ocurre en la práctica y lo que surge de documentos o acuerdos, debe darse preferencia a lo primero es decir, a lo que sucede en el terreno de los hechos..."*

Figura 6.5: Ejemplos referencias a Doctrinos

Al terminar el proceso de selección de fuentes sobre las 1000 sentencias, se obtuvo una lista de 74 nombres correspondientes a Doctrinos. Más allá de los resultados satisfactorios obtenidos en esta etapa, sería ideal prolongar este trabajo para obtener más nombres y referencias a Doctrinos. Sin embargo, esto implicaría resolver muchos problemas relacionados con este tema los cuales están por fuera del alcance de este trabajo, como por ejemplo determinar que dos nombres se refieren a la misma persona. Un ejemplo de esto se dio con nombres como "Gamarra" y "Enrique Gamarra", los cuales corresponden a la misma persona. Una alternativa sería mantener únicamente el apellido, pero se induciría a más error que el que se genera al mantener los dos nombres en la lista. Además, el nombre también podría aparecer de la forma "Gamarra, Enrique" por lo que se deberían aplicar otros métodos para reconocer el nombre completo.

6.1.4. SPIED: Stanford Pattern-based Information Extraction and Diagnostics

Luego del proceso descripto anteriormente, el cual sirvió para obtener una lista de doctrinos, surgió la necesidad de experimentar con un sistema que brinde mayor nivel de automatización. Para esto, se pensó en un sistema que aprendiera, a partir de algunos ejemplos, los diferentes patrones de texto en los cuales se encuentran las referencias a doctrinos. Luego de investigar se llegó a un sistema desarrollado por el grupo de PLN de la Universidad de Stanford (Gupta et al. [32]), el cual tiene como objetivo la extracción de información basada en patrones, para la cual utiliza Bootstrapping, y brindar una interfaz amigable para poder analizar tanto los patrones aprendidos como las entidades extraídas. Cabe destacar que en este trabajo, se utilizó únicamente el sistema de extracción.

Bootstrapping es un método que parte desde una lista inicial de entidades conocidas, e iterativamente analiza el texto en el cual estas entidades ocurren, aprende nuevos patrones y por ende nuevas entidades que cumplan dichos patrones en un texto no etiquetado. Contar con una herramienta de extracción basada en patrones, y debido a que estos resultan fáciles de interpretar para las personas, es posible identificar fuentes de errores, observando patrones responsables de extraer entidades incorrectas y viceversa, para luego corregirlos. Estos patrones son puntuados por su capacidad de extraer entidades más positivas y entidades menos negativas. Un problema es que debido a la falta de datos etiquetados, el sistema asume que las entidades no etiquetadas son negativas o son ignoradas por las medidas de puntuación de patrones existentes.

En Gupta et al. [32], se presenta el proceso iterativo que sigue el sistema para el reconocimiento de patrones y entidades. En este caso, el proceso es el siguiente:

I) Etiquetado de datos: se etiqueta el texto utilizando, en primera instancia, la lista inicial de Doctrinos conocidos y luego las entidades aprendidas durante las iteraciones.

II) Generación de patrones: una vez etiquetado el texto con los ejemplos positivos, se utiliza el contexto de los mismos para el proceso de creación de patrones candidatos. Es decir, se identifican los patrones en los cuales los nombres de los Doctrinos aparecen.

III) Aprendizaje de patrones: los patrones candidatos son puntuados según una medida definida y los que obtengan mayor puntuación son agregados al conjunto de patrones aprendidos. La cantidad máxima de patrones a tomar como correctos es configurable mediante un parámetro.

IV) Aprendizaje de Doctrinos: los patrones seleccionados en el punto anterior son aplicados al texto para obtener Doctrinos candidatos. Una vez obtenidos, estos son puntuados y, al igual que con los patrones, los que posean mayor puntuación son agregados al conjunto de entidades correctas. Existe un parámetro donde se configura la cantidad máxima de entidades a agregar al conjunto.

V) Se repiten los pasos anteriores por una determinada cantidad de iteraciones o hasta que no hayan más patrones o entidades para aprender.

En cuanto a la medida de puntuación de entidades, el sistema utiliza la indicada en Gupta et al. [33], la cual consiste en el promedio de varias medidas calculadas para la entidad. Una de estas medidas tiene en cuenta la distancia de Damerau-Levenshtein [34] de la entidad candidata sobre las entidades correctas,

mientras que otra utiliza el mismo mecanismo pero aplicado sobre las entidades negativas. El algoritmo Damerau-Levenshtein se caracteriza por mejorar al algoritmo simple de Levenshtein ya que agrega la operación de transposición de caracteres al conjunto de operaciones compuesto por Inserción, Sustitución y Eliminación. Otras medidas se basan en la similitud de los contextos en los cuales ocurre la entidad candidata y las entidades negativas y positivas, agrupando las entidades cuyos contextos sean más similares. Dado esto, si alguna entidad sin etiquetar se agrupa con entidades positivas, recibirá una puntuación más alta que las agrupadas con entidades negativas.

Por otro lado, el sistema permite variar el método de puntuación de patrones mediante el parámetro "patternScoring". En este caso se realizaron pruebas con los valores "RatioAll" y "SqrtRatioAll" debido a que son los utilizados por los autores en trabajos previos. Otros parámetros utilizados en la configuración del sistema fueron:

- numWordsToAdd = 5. Cantidad máxima de palabras a extraer en cada iteración.
- numPatterns = 5. Cantidad máxima de palabras a extraer por cada patrón.
- numIterationsForPatterns = 8. Cantidad máxima de iteraciones.
- tokenize.language = es. Idioma de tokenizado.
- usePreviousContext = true. Utilizar texto anterior a la entidad como contexto.
- useNextContext = true. Utilizar texto posterior a la entidad como contexto.
- minWindow4Pattern = 2. Tamaño mínimo de ventana de contexto.
- maxWindow4Pattern = 4. Tamaño máximo de ventana de contexto.

Este sistema se probó con 100 sentencias aleatorias extraídas del corpus de la BJN. Si bien se experimentó con diferentes configuraciones de parámetros, no se pudo obtener un resultado de valor, debido a que en todas las pruebas realizadas se lograba obtener una lista de nombres la cual incluía algunos Doctrinos pero también muchos otros nombres propios pertenecientes a personas participantes de la sentencia. Estos resultados pueden deberse a varios factores, como por ejemplo la introducción de entidades incorrectas en una iteración. Esto afecta al rendimiento total del sistema debido a que la entidad incorrecta es marcada e incluida dentro del conjunto de entidades correctas, lo cual afecta a las puntuaciones de iteraciones futuras. Otro punto que explica los resultados es la gran similitud entre los contextos en los cuales se encuentran los nombres propios, ya sean de Doctrinos o no. Finalmente, un gran problema enfrentado al utilizar SPIED es sin dudas la dificultad de configuración del sistema, ya que por defecto éste viene configurado para procesar textos en inglés y utiliza una gran cantidad de parámetros y herramientas dependientes del idioma.

6.2. Análisis Semántico

En esta sección se trata el proceso necesario para el etiquetado semántico de referencias. Esto es, poder identificar el motivo por el cuál, en una sentencia, se realizó una citación a un determinado objeto legal. Para poder lograr este objetivo primero es necesario contar con las referencias identificadas, lo cual está contemplado en el capítulo 5. Luego, siguiendo la línea de Sadeghian et al. [7], es necesario analizar las particularidades del contexto de las mismas, más precisamente de su predicado, para poder así determinar la razón por la cual se realizó la referencia.

Los autores definen al predicado como el mayor segmento de texto que exprese directamente la razón por la cual se encuentra presente la referencia, no estando este relacionado con ningún tema en específico y manteniendo su sentido si se lo coloca junto a otra referencia. Luego, con información acerca de los predicados, y mediante la utilización de diferentes métodos, es posible realizar una categorización o etiquetado semántico de las citaciones en base a un conjunto de categorías resultante de un análisis manual de las sentencias de la BJN.

Poder contar con la razón por la cual se realizó una citación a un objeto legal resulta útil en varios escenarios del área de estudio. Por ejemplo, se podría analizar cuáles de estos son frecuentemente referenciados en conjunto y por qué razones, para luego definir la relación entre los mismos. Otra aplicación podría ser al momento de modificar un artículo de algún Cuerpo Normativo, para poder ver el impacto del mismo en futuras sentencias. Es decir, analizar las razones de su utilización en sentencias anteriores y poder prever los cambios en su utilización a partir de la modificación.

Cabe destacar que luego de realizar un análisis manual de los predicados de las referencias sobre un conjunto de sentencias de la BJN, no fue posible extraer información significativa sobre los predicados y el sentido semántico de las citaciones. Debido a esto, se concluyó que esta etapa supera los alcances definidos en este proyecto y amerita un trabajo aparte. De igual forma, a pesar de no haber proseguido, se deja a continuación una guía que abarca diferentes puntos para lograr los resultados esperados.

6.2.1. Extracción de predicados

La idea en esta etapa es poder crear un método que sea capaz, a partir del texto de una sentencia y las referencias contenidas en el mismo, de extraer los predicados asociados a cada una de ellas. Para esto, los autores utilizaron CRF al cual entrenaron con distintos ejemplos anotados manualmente por un conjunto de especialistas en el área. Debido a la falta de este recurso para las sentencias de la BJN, en un comienzo se procedió a utilizar la herramienta BRAT para realizar anotaciones manuales de predicados de referencias. Sin embargo, como se mencionó anteriormente, la mayoría de las sentencias contaban con muy pocas o casi nulas citaciones con predicado explícito.

Si se contara con un conjunto de ejemplos anotados, el siguiente paso consistiría en enmascarar las referencias extraídas con un token diferente a los encontrados en las sentencias para que no interfieran en el entrenamiento del CRF y para que puedan ser tomados como parte de los atributos definidos para cada token. Luego, se podría reutilizar el conjunto de atributos propuesto por los autores, el cual consiste en:

- El token, el token anterior y el siguiente en minúscula.
- El POS-Tag del token y sus dos vecinos.
- Las primeras dos letras y las últimas dos letras de los POS-Tags.
- Cinco atributos booleanos asociados a la distancia entre el token y la referencia.
    1. Si el token ocurre luego de la referencia.
    2. Si no existen otros tokens entre la referencia y la palabra.
    3. Si hay únicamente un token entre la referencia y la palabra.
    4. Si hay 2 o más tokens entre la referencia y la palabra.
    5. Si hay 4 o más tokens entre la referencia y la palabra.
- Valor booleano que indica si el token es numérico.
- Valor booleano que indica si el token está al comienzo de la oración.
- Valor booleano que indica si el token está al final de la oración.
- Valor booleano que indica si el token es un signo de puntuación.

Los ejemplos I) a III) de la figura 6.6 muestran el patrón más común de referencias encontrado en las sentencias de la BJN. En estos casos el texto de las referencias se encuentra entre paréntesis como adición al texto de la oración. Es decir, el contenido semántico de la referencia se encuentra implícito asumiendo que el lector tiene un determinado conocimiento acerca del tema sobre el que trata el objeto legal referenciado. En el contexto de este trabajo, no se cuenta con dicha información, por lo cual resulta imposible tener en cuenta a dichos ejemplos para la determinación del predicado. Este mismo problema es mencionado en [7], donde se aclara que es importante tener acceso al contenido del cuerpo legal referenciado para poder deducir completamente la semántica. Para ejemplificar mejor esta situación, se adjunta el segmento de texto correspondiente al numeral 5 del artículo 46 del Código Penal presente en el ejemplo I): "*Atenúan el delito cuando no hubieran sido especialmente contempladas por la ley al determinar la infracción, las siguientes: ... 5. (Minoría de edad). La edad, cuando el agente fuere menor de veintiún años y mayor de dieciocho.*". Este artículo lista algunos escenarios en los cuales la pena de un delito se puede atenuar, dentro de los cuales se encuentra el de "minoría de edad". Teniendo esta información se puede deducir que, a lo que se está haciendo referencia en el texto de la citación, es que no se tendrá en cuenta lo indicado por dicho artículo y se tomarán los delitos anteriores sin ninguna atenuación.

I) "Malgrado la minoridad relativa del agente **(art. 46 num. 5º CP)**, la consideración de los antecedentes del encausado se impone, porque..."

II) "Por cierto que interesa al colectivo social el acceso a la Justicia, que tiene expresión concreta en el derecho a acudir ante los tribunales **(arts. 11 C.G.P.)** acercar las sedes de los tribunales a toda la población **(art. 22.1)** procurar la especialización **(art. 22.4 C.G.P.)**"

III) "Público, es el único motivo para la sucumbencia que encontradamente pretenden las partes; los agravios de la Defensa carecen de fundamento **(arts. 253, 255 CPP)**."

IV) "...resulta claro que al presentarse dicho petitorio el día 24 de febrero del corriente año había ya transcurrido largamente el término establecido en el **segundo inciso del art. 4º de la Ley 16011**."

V) "Conforme con lo dispuesto por el **art. 508 del Código General del Proceso**, se podrá promover la declaración de inconstitucionalidad siempre que deba aplicarse una Ley o una norma que tenga fuerza de Ley."

Figura 6.6: Ejemplos predicados

Si bien en muchos casos, sabiendo de qué trata el objeto legal referenciado, se puede deducir la semántica asociada a la citación, existen otros en los cuales esta información no es suficiente para que una persona ajena al ámbito legal pueda hacerlo. Un ejemplo de esto se da con la referencia número III), en la cual se mencionan los artículos 253 y 255 del Código del Proceso Penal, cuyas redacciones presentadas a continuación, resultan difíciles de relacionar con el contexto en el cual se están citando.

- Artículo 253: "(Recursos). Cuando la resolución ordene la medida solicitada u otra similar, la misma será apelable sin efecto suspensivo.".
- Artículo 255: "El proceso de conocimiento comprende la primera y la segunda instancia y el recurso de casación.".

Luego, los puntos IV) y V) representan a la mayoría de los predicados que ocurren en las referencias explícitas, ya que estos definen a la semántica de la citación como apoyo en la argumentación. Es decir, la mayoría de las referencias se realizan como parte de la argumentación de una decisión tomada. A continuación se listan otros de los predicados más comunes encontrados en las sentencias de la BJN, los cuales siguen la misma línea de estos ejemplos.

- según lo previsto
- conforme al
- perceptuado por
- determinado en
- expuesto en

Dadas estas observaciones, resulta difícil la definición de un conjunto de etiquetas semánticas posibles, debido a que dentro de los ejemplos utilizables, la gran mayoría serían clasificados con la etiqueta "base legal". Esta etiqueta surge del conjunto utilizado por los autores, y es empleada cuando se referencia a un objeto legal como base de una argumentación. Este conjunto fue creado a partir de análisis realizados por diferentes profesionales del área legal. Por lo tanto, tomando en cuenta los puntos mencionados anteriormente, y sumado a la falta de recursos en el ámbito legal disponibles para colaborar en un análisis profundo de la semántica presentada en las referencias, así como a la falta de conocimiento sobre el área, se decidió no proseguir con esta etapa.

6.2.2. Etiquetado Semántico

En el caso en el que se pudieran obtener los predicados, estos se utilizarían como parte del conjunto de entrenamiento del método de clasificación semántica. Los autores utilizaron tanto el enfoque de aprendizaje supervisado como el de aprendizaje no supervisado. En los dos casos utilizaron Word Embeddings obtenidos a partir de un corpus de texto extraído de Wikipedia y re entrenados con el Código de Estados Unidos para ajustar los vectores al área de interés de su trabajo [35]. Luego, definieron un vector asociado a cada predicado, el cual se calcula con el promedio de los vectores de las palabras que están incluidas en el mismo.

El método seleccionado para aprendizaje no supervisado fue K-Means, tomando una cantidad de centroides mayor a la cantidad de etiquetas definidas en su conjunto, para de esta forma poder capturar mejor los casos en los cuales la etiqueta no está clara y existe superposición de clusters. Finalmente, a todos los ejemplos de un cluster se los clasificaba con la etiqueta del punto más cercano al centroide. Con este método probaron que es posible obtener buenos resultados en aquellos casos en los cuales los predicados demuestran explícitamente su sentido semántico, y en el cual el conjunto de etiquetas esté bien definido y sea disjunto. En este trabajo quedó demostrado que es difícil en el área legal cumplir con estas hipótesis.

Para el enfoque de aprendizaje supervisado, plantean la utilización de dos métodos, Support Vector Machine (SVM) y Regresión Logística. En este caso modelaron al problema como un problema de clasificación multiclase, tomando dos estrategias de evaluación, la habitual considerando la etiqueta más probable y otra tomando las dos etiquetas más probables determinadas por el clasificador. Esto último significa que al momento de evaluar, se toman como correctos los casos en los cuales la etiqueta anotada sea una de las dos más probables para el clasificador. Con el primer enfoque obtuvieron un accuracy de 64 % para SVM, mientras que con el segundo alcanzaron un valor de 79 %.

Capítulo 7

Grafo de referencias

El siguiente paso luego de culminada la etapa de extracción de referencias, consistió en crear una base de datos de grafos de manera de almacenar la información de estas referencias y sus relaciones. Estas bases de datos se caracterizan por utilizar estructuras de grafos para representar la información, es decir, representan los datos en forma de nodos conectados por aristas, donde estas representan las relaciones entre los diferentes datos. Esta forma de representación favorece a aquellos escenarios en los cuales los elementos que componen la información están muy relacionados, debido a que la estructura de grafos facilita la interpretación de los mismos. Además, este tipo de bases de datos permite aprovechar todas las propiedades que brinda la teoría de grafos, generando grandes beneficios a la hora de realizar análisis y búsquedas sobre los datos.

7.1. Características

Flexibilidad del modelo de datos

Una de las principales características que presenta esta forma de almacenamiento consiste en la gran flexibilidad que otorga al usuario. Esto se debe a que no es necesario tener un modelo de datos a utilizar fijo y pre-definido, sino que se permite expandir el modelo en cualquier momento para agregar nuevos nodos y relaciones. Esta flexibilidad también lleva a que no se presente el problema de los valores nulos como en el caso de las bases de datos relacionales, debido a que se brinda soporte para que nodos del mismo tipo puedan tener diferentes atributos. Por ejemplo, un nodo del tipo “Persona” puede tener los atributos “nombre” y “edad”, mientras que otro nodo del mismo tipo puede no tener el atributo “edad” (debido a que no se cuenta con dicha información) pero tener el atributo “dirección”. Otra característica importante relacionada con la flexibilidad está dada por el tamaño variable de los registros, lo cual evita problemas al ingresar datos de tamaños muy diferentes así como también ahorra el trabajo de definir el tamaño y tipo de los atributos.

Estas características tienen un impacto positivo para la productividad del desarrollador y el riesgo del proyecto. La posibilidad de ir variando el modelo de datos según los cambios que requiere el área en la cual se está trabajando también significa que se tiende a realizar menos migraciones, generando una reducción en los gastos generales de mantenimiento y riesgos.

Performance

Las bases de datos de grafos permiten realizar consultas complejas de manera eficiente debido a que las relaciones entre los datos están definidas a nivel estructural. Esto es una ventaja con respecto a las bases de datos relacionales ya que estas deben realizar diferentes joins para buscar las relaciones entre los datos, lo cual se realiza cada vez que se ejecuta una consulta. Mientras que el tiempo consumido por estos joins se incrementa a medida que la cantidad de datos aumenta, los tiempos de las consultas en las bases de datos de grafos tienden a mantenerse constantes. Como resultado, el tiempo de ejecución para cada consulta es proporcional únicamente al tamaño del subgrafo recorrido para satisfacer la consulta, en lugar del tamaño del grafo completo.

Intuitivo

Debido a su naturaleza conectiva y dirigida, la representación de los datos y sus relaciones en forma de grafo resulta mucho más intuitiva que los datos almacenados en tablas. Las relaciones en un grafo naturalmente forman caminos, por lo tanto, realizar consultas en una base de datos de este tipo implica seguir caminos entre los nodos. Estos caminos no solamente se adaptan a la forman en la que las personas piensan en cómo están relacionadas las cosas, sino que también facilitan de alguna manera contestar el tipo de preguntas que usualmente se le desean hacer al dominio en el cual se está trabajando.

Teoría de grafos

Como se mencionó anteriormente, con este enfoque de manejo y almacenamiento de datos se pueden aprovechar las propiedades de los grafos, ya sea para realizar consultas más eficientes así como para analizar los datos almacenados. En cuanto al análisis de los datos, las funcionalidades más útiles con las que se cuentan se dividen en cuatro grupos de algoritmos:

- Búsqueda de caminos: conjunto de algoritmos para encontrar el camino más corto o evaluar la disponibilidad y calidad de distintos caminos.
- Centralidad: conjunto de algoritmos para determinar la importancia de un nodo o conjunto de nodos en un grafo.
- Detección de comunidades: conjunto de algoritmos para estudiar cómo están agrupados los nodos, de qué forma están particionados los distintos grupos y su tendencia a agruparse o desagruparse.
- Similitud: conjunto de algoritmos para estudiar la similitud entre diferentes nodos.

7.2. Herramienta

Para el almacenamiento de los datos se optó por la utilización de la herramienta Neo4j, la cual se encuentra implementada en Java y cuenta con una versión OpenSource. Esta herramienta cuenta con APIs para varios lenguajes populares como lo son Python, .NET y Java, permitiendo construir aplicaciones e integrar de manera simple y eficiente el sistema de base de datos de grafos. La herramienta utilizada en este caso fue la librería de Python Py2Neo.

Aplicación

En el marco del análisis de sentencias de la BJN, se decidió utilizar esta herramienta para la administración de los datos relacionados con las distintas referencias a objetos legales. Como se mencionó en la sección anterior, las bases de datos orientadas a grafos presentan muchas ventajas las cuales pueden ser aprovechadas en el área de las referencias en textos legales.

Para esto, se definieron siete tipos de nodos con distintos atributos los cuales pueden estar presentes tanto como atributos del nodo como atributos de una relación con el mismo, siempre y cuando se haya podido extraer dicha información en el proceso de extracción de las referencias.

- Articulo: se tuvieron en cuenta su número, numeral, literal e incisos relacionados al número en los casos que correspondiera, y su fuente. Un ejemplo se ve en "artículo 37.2 literal a) del C.G.P, donde el número es 37, el numeral 2, el literal A y la fuente el "Código General del Proceso".
- Sentencia: se tuvieron en cuenta su número, el año, la sede (tribunal o juzgado) a la que perteneciera y el turno en los casos en los que correspondiera. En la referencia "sentencia 168/2017, Tribunal de Apelaciones en lo Civil de 2do Turno", el número es 168, el año 2017, la sede "Tribunal de Apelaciones en lo Civil" y el turno 2.
- Decreto: se tuvieron en cuenta su número y año. "Decreto 611/980" posee como propiedades al número 611 y al año 1980.
- Ley: se tuvo en cuenta únicamente el número debido a que muy pocas veces éstas eran referenciadas agregando la fecha correspondiente. Por ejemplo "Ley 19090", aunque también puede presentar el nombre que la identifica (por ejemplo: "Ley Orgánica Policial").
- Ficha: se tuvieron en cuenta su número, y su año. La mayoría de las referencias a estos objetos legales se da en el formato IUE (Identificador Único de Expediente), es decir "Sede-NroRegistro/Año", por ejemplo "0002-002487/2015", donde se toma como año al 2015 y por simplicidad se toma como número a 0002-002487.
- Doctrino: se tuvo en cuenta únicamente su nombre completo.
- Cuerpo Normativo: este tipo de nodo se agregó con la finalidad de unificar las referencias a objetos legales pertenecientes al mismo cuerpo normativo. El objetivo es que exista un único nodo representando a por ejemplo la "Constitución de la República", y que cada arista entrante al mismo provenga de un objeto legal perteneciente a dicho Cuerpo Normativo.

Luego, las aristas entre los nodos representan el tipo de relación entre estos. Estas relaciones se clasifican en los siguientes tipos:

- Referencia: esta relación se utiliza entre dos nodos A y B para indicar que desde el objeto legal A se hace referencia al objeto legal B. Un ejemplo de esta relación se da cuando en el texto de una determinada sentencia, se hace referencia a por ejemplo un artículo del "Código General del Proceso". En dicho caso, la sentencia sería el nodo A, y tendría una arista saliente hacia el nodo B, el cual representaría al artículo mencionado. Para ejemplificar se puede tomar como referencia a la

redacción de la Sentencia mencionada anteriormente, dentro de la cual se encuentra el segmento de texto *"...supuestos exigidos por el artículo 125 del Código del Proceso Penal..."*. Por lo tanto el nodo A no sólo contendrá la información de la sentencia 168/2017, si no que también tendrá una conexión con un nodo B, correspondiente al artículo 125 del C.G.P. El objetivo para etapas futuras de este trabajo, es que en dicha relación se especifique la razón semántica por la cual se dio la referencia.

- Fuente: este tipo se utiliza para relacionar a un artículo con su fuente, la cual puede ser tanto un Cuerpo Normativo, una Ley o un Decreto. En este caso el nodo A sería el correspondiente al artículo 125, y tendría una arista saliente hacia el nodo correspondiente al "Código General del Proceso".
- Expediente: este tipo de relación se utiliza para explicitar la relación entre una sentencia y su expediente, el cual está representado con un nodo de tipo Ficha. Un ejemplo de esta relación se da entre la sentencia 168/2017 mencionada anteriormente y su expediente, la ficha "103-269/2016", la cual se correspondería con el nodo B.

Lenguaje de consulta

Aunque existen varios lenguajes de consulta sobre bases de datos de grafos, el utilizado por la herramienta Neo4j es Cypher. Este es un lenguaje declarativo que permite consultas y actualizaciones expresivas y eficientes en una base de datos orientada a grafos. El ser un lenguaje de consulta expresivo y a su vez compacto, sumado al hecho de ser el más ampliamente implementado, lo convierte en el estándar de facto. Además, sintácticamente tiene varias similitudes con SQL por lo que resulta más fácil de utilizar.

La forma de accionar de este lenguaje consiste en una búsqueda por patrones. Es decir, Cypher permite que un usuario (o una aplicación que actúe en nombre de un usuario) realice una búsqueda sobre datos que coincidan con un patrón específico.

A continuación se muestra un ejemplo de consulta sobre el grafo de referencias, la cual tiene como objetivo obtener aquellos artículos pertenecientes al "Código General del Proceso", los cuales fueron referenciados en la Sentencia número 741 de 2008. Esta consulta expresada en Cypher sería la siguiente:

```
MATCH (a:Articulo)-[:Fuente]->(c:CuerpoNormativo),
(s:Sentencia)-[:Referencia]->(a)
WHERE c.nombre = 'Código General del Proceso' AND s.numero = 741 AND s.año = '2008'
RETURN a,c,s
```

Como se puede ver, la sintaxis de la consulta resulta bastante intuitiva y se asemeja en buena forma a lo que se desea extraer. En la imagen 7.1 se puede ver que el resultado de dicha consulta genera un subgrafo el cual consiste en el nodo correspondiente al cuerpo normativo mencionado (nodo de color marrón) junto a los nodos correspondientes a los artículos pertenecientes al mismo (nodos rojos), y al nodo correspondiente a la sentencia 741 (nodo anaranjado).

Figura 7.1: Resultado consulta Cypher en Neo4j

Extensiones y librerías

Como se mencionó anteriormente, gracias a la teoría de grafos, se pueden aplicar diversos algoritmos sobre los mismos, pudiendo generar análisis tanto sobre los datos como sobre las relaciones. Neo4j presenta varias extensiones y librerías las cuales están centradas en resolver de manera única y simple estos algoritmos para el usuario. Por ejemplo, la librería Graph Algorithms está diseñada para implementar diferentes algoritmos de grafos de manera eficiente. Estos pueden proporcionar información sobre entidades relevantes en el grafo (centralidades, clasificación) o estructuras inherentes como las comunidades (detección de comunidad, división de grafos, agrupación).

Muchos algoritmos de grafos son enfoques iterativos que frecuentemente recorren el grafo para el cálculo utilizando tanto recorridos aleatorios, búsquedas en amplitud o en profundidad, o coincidencia de patrones. En los casos en los que el grafo toma dimensiones muy grandes, se genera un crecimiento exponencial de los posibles caminos así como un aumento en la distancia de los mismos, causando que muchos de los enfoques presenten una alta complejidad algorítmica. En el caso de la librería de Neo4j mencionada anteriormente, se poseen implementaciones optimizadas de los algoritmos que utilizan ciertas estructuras del grafo, memorizan partes ya exploradas y paralelizan operaciones. Algunos de los algoritmos que brinda esta librería son:

- PageRank: algoritmo utilizado para estudiar la importancia de los nodos, teniendo en cuenta la cantidad de referencias que se realizan sobre él así como la importancia de los nodos que lo referencian.
- Betweenness Centrality: algoritmo utilizado para detectar la cantidad de influencia que un nodo tiene sobre el flujo de información en un grafo. Usualmente se usa para encontrar nodos que sirvan como conexión de una parte de un grafo a otra.
- Eigenvector Centrality: algoritmo utilizado para medir la influencia transitiva o la conectividad de los nodos. Las relaciones con los nodos de alta puntuación contribuyen más a la puntuación de un nodo que las conexiones a los nodos de baja puntuación. Una puntuación alta significa que un nodo está conectado a otros nodos que tienen puntuaciones altas.
- Same Community algorithm: algoritmo utilizado para determinar si dos nodos pertenecen a la misma comunidad. Si esto sucede, existe una mayor probabilidad de que haya una relación entre ellos en el futuro, si aún no existe.

7.3. Proceso de carga de datos

Luego de culminadas las etapas anteriores, se contaba con un modelo entrenado capaz de extraer segmentos de texto correspondientes a referencias a objetos legales con una medida micro-F1 de 87.63 % (5.10). Al contarse únicamente con el segmento completo de texto, se requirió de un post procesamiento para distinguir las distintas partes que conforman a una referencia. Estas partes dependen principalmente del tipo de objeto legal al cual se hace referencia, incluyendo por ejemplo número y fecha. Las propiedades de los objetos legales tenidas en cuenta son las descriptas en 7.2.

El proceso general de carga de datos consistió entonces en desarrollar un sistema que se encargue de extraer las referencias existentes en un texto legal de la base de BJN, las procese obteniendo las propiedades más importantes según su tipo y luego adapte dicha información a una consulta en lenguaje Cypher para así poder crear los nodos y relaciones correspondientes en la base de datos a través de la librería Py2Neo.

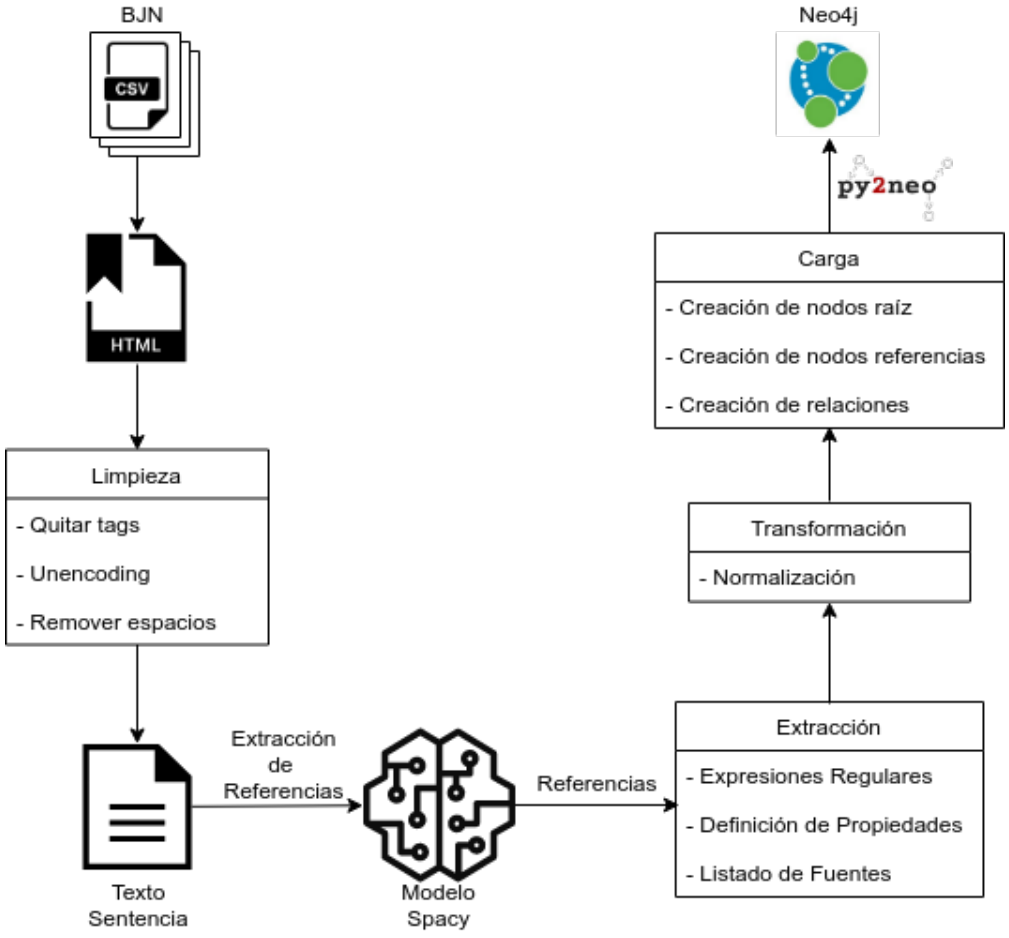

Figura 7.2: Flujo del sistema

7.3.1. Extracción de propiedades

El problema de extraer las propiedades de cada tipo de referencia se abordó de forma individual, debido a que gracias al modelo entrenado descripto en la sección 5.3 se contaba con la información de a qué tipo pertenecía cada referencia obtenida pudiendo aprovechar así las particularidades de las mismas. Estos procesos se realizaron de forma manual, reutilizando en muchos casos lo realizado mediante expresiones regulares debido a que, como se mencionó en la sección 4.1, estas resultan de gran utilidad cuando no se cuenta con ejemplos anotados.

Ficha

El proceso para extraer el número y año de las referencias a Fichas consistió en aplicar una expresión regular que reconociera a los números seguidos de un año o una fecha completa. Para esto se reutilizaron parcialmente las expresiones regulares creadas en la primera etapa del proyecto. Cabe destacar que al saber que la referencia se trata de una Ficha, no es necesario reconocer las palabras clave asociadas a las mismas, sino que alcanza con reconocer el número y año. Esto facilitó el trabajo en comparación con el proceso de expresiones regulares descripto en la sección 4.1 debido a que para dicha etapa únicamente se contaba con el texto plano de las sentencias. A continuación se pueden ver algunos ejemplos de referencias a Fichas extraídas por el modelo:

I) `IUE 32-22/2003`

II) `FICHA 2-63793/05`

III) `Ficha N 2-46.871/2010`

IV) `Ficha 78/103/2000`

V) `ficha I.U.E.1-541/2003`

VI) `F.N179/626/07`

Figura 7.3: Ejemplos referencias a Fichas

Un requisito para que se mantenga un nivel de consistencia y coherencia suficiente en la base de datos de grafos, era que no se dupliquen nodos correspondientes a una misma instancia de un objeto legal. En el caso de las Fichas, para solucionar este problema y homogeneizar el formato en el cual los datos eran cargados en el grafo, se trabajó teniendo en cuenta los diferentes grupos contenidos en la expresión regular. Se definieron 4 grupos:

- Grupo 1: Número completo. Si este grupo contenía alguna coincidencia, en él se encontraba el número completo de la Ficha, incluyendo símbolos separadores. Por ejemplo en el caso de la referencia I) de 7.3, el grupo 1 obtiene una coincidencia en “32-22”. Este grupo se utiliza únicamente en los casos en los que la ficha no se encuentra en formato IUE, quedando el grupo 2 o grupo 3 sin coincidencias.
- Grupo 2: En el caso en el que la referencia a la Ficha estuviera en formato IUE, este grupo es el que obtiene la primera parte del número de la misma. Por ejemplo en el caso de la referencia II) este grupo contendría al número “2”.

- Grupo 3: En el caso en el que la referencia a la Ficha estuviera en formato IUE, este grupo es el que obtiene la segunda parte del número de la misma. Por ejemplo en el caso de la referencia III) este grupo contendría al número "46.871".
- Grupo 4: Este grupo obtiene el año de la Ficha, ya sea completo, de dos dígitos o de tres dígitos. En los ejemplos I) a VI) de 7.3 este grupo almacenaría los valores "2003", "05", "2010", "2000", "2003" y "07" respectivamente.

Ley

En cuanto a las referencias a Leyes, el proceso fue muy similar al realizado para las Fichas. En este caso, la única propiedad de interés era el número de Ley. Para esto se utilizó una expresión regular que tuviera en cuenta los números con y sin punto, por ejemplo 13578 y 13.578. Se reconocieron dos grupos, el primero en el cual se almacenaba la primera o las primeras dos cifras del número y el segundo en el cual se almacenaban las cifras restantes.

Decreto

Para las referencias de tipo Decreto, resultó de interés obtener tanto su número como su año. Debido a que en el proceso de anotación descripto en la sección 4.1.2 no se distinguieron los decretos separados por ",", el modelo entrenado no lograba distinguir entre varios decretos dentro de una misma referencia y clasificaba a un segmento de texto con referencias a varios decretos como un único Decreto. Es por esto que fue necesario separar estos decretos para poder aplicarle una expresión regular a cada candidato de número y año. Por ejemplo el segmento de texto "decretos N 332/992, 578/94, 217/001, 281/01 y 33/02", se separaba en "decretos N 332/992", "578/94", "217/001", "281/01" y "33/02", y luego se le aplicaba una expresión regular con cuatro grupos:

- Grupo 1: Primera o primeras dos cifras del número. En el primera segmento de texto este grupo almacenaría "33".
- Grupo 2: Siguientes cifras del número. En el primer segmento de texto este grupo almacenaría "2".
- Grupo 3: Año con 4 dígitos en fecha separada. Si este grupo tenía coincidencia significaba que el año surgía de una descripción explícita de la fecha, por ejemplo: "357 de 25 de marzo de 2008". En este caso el grupo obtendría la fecha 2008.
- Grupo 4: Año compuesto. Este grupo obtiene coincidencia cuando el año viene junto al número ya sea completo o con dos o tres dígitos. Por ejemplo: "992" en el primer ejemplo o "02" en el último.

Sentencia

Como se mencionó en reiteradas ocasiones, para que una Sentencia sea identificada completamente se requiere saber su número, año, sede, y turno. Es por esto que el proceso de extraer dichas propiedades de las referencias de este tipo se dividió en diferentes partes. Al igual que en el caso de los Decretos, algunos segmentos de texto clasificados como Sentencia contenían varias referencias separadas por la conjunción coordinante "y" y el signo de puntuación ",". En primera instancia, se procedió a dividir las mismas teniendo

en cuenta los criterios mencionados anteriormente, para luego analizarlas individualmente.

I) `sentencias Nos. 111/04, 177, 234, 314/07`

II) `N 31/09 TRIBUNAL DE APELACIONES EN LO CIVIL DE SEPTIMO TURNO`

III) `Sentencia No. 256/97 de la Suprema Corte de Justicia`

IV) `Sent. No. 27 de 21/3/2000 dictada por el T.A.P. 3`

V) `Sentencia definitiva de primera instancia No. 22, dictada el 8 de marzo de 2007 por el Juzgado Letrado de Ciudad de la Costa de 4º Turno`

VI) `N. 96/2006 dictada por la Sra. Juez Letrado de Primera Instancia de Treinta y Tres de 2do.Turno`

Figura 7.4: Ejemplos referencias a Sentencias

Luego, a cada referencia se le aplicó una expresión regular para determinar el turno relacionado a la sede que la emitió. Para reconocer a un turno se tuvieron en cuenta aspectos como la aparición en el texto de esta palabra o una de sus derivadas, la aparición del símbolo "º" así como la aparición de una sigla correspondiente a un Tribunal como se ve en el ejemplo IV) de 7.4 con "T.A.P 3". Posteriormente, se procedió a reconocer el número y el año de la Sentencia. Para esto se utilizó una única expresión regular la cual consiste en tres grupos:

- Grupo 1: Número completo de la Sentencia. Por ejemplo "256", "96" en los ejemplos III) y VI) respectivamente.
- Grupo 2: Año explícito de la Sentencia, es decir cuando en la referencia se encuentra la fecha completa separada del número. Este grupo almacenaría por ejemplo los años "2000" y "2007" en los ejemplos IV) y V) respectivamente.
- Grupo 3: Año compuesto, es decir cuando el año se encuentra junto al número de la Sentencia separado únicamente por algún símbolo. En los ejemplos II) y III) este grupo almacenaría los números "09" y "97" respectivamente.

Aunque suena razonable que el número y el año podrían ser extraídos en primera instancia debido a que sin ellos una sentencia no estaría ni parcialmente definida, esto no pudo ser así debido a que el número de turno generaba confusiones a la hora de la aplicación de la expresión regular. Una opción hubiera sido complejizar dicha expresión, pero la opción tenida en cuenta finalmente consistió en reconocer primero al turno, y en caso de que este estuviera explícito en la referencia, proceder a enmascararlo para que no sea interpretado como parte del número o del año. Por ejemplo, luego de este paso el ejemplo IV) con su turno enmascarado quedaría como "Sent. No. 27 de 21/3/2000 dictada por el T.A.P. **X**".

Debido a la gran ocurrencia de referencias compuestas como por ejemplo la referencia del punto I) presentada en los ejemplos anteriores, y sumado a la necesidad de separar las mismas para su análisis, surgió el inconveniente de que muchas de estas quedaban aisladas de su fecha. Esto, como se puede ver en el ejemplo mencionado, surge debido a que cuando se referencia a varias sentencias del mismo año, éste únicamente

es colocado explícitamente como parte de la referencia a la última sentencia del segmento. En el ejemplo, las sentencias número 177 y 234 se corresponden al año 2007, el cual únicamente aparece junto al número de sentencia 314. La solución a este problema consistió en invertir el orden de las sentencias provenientes de la separación por "y" y por "," e ir almacenando el último año reconocido para luego relacionarlo con aquellos números aislados anteriores en el orden regular, proceso que se puede apreciar en la siguiente imagen.

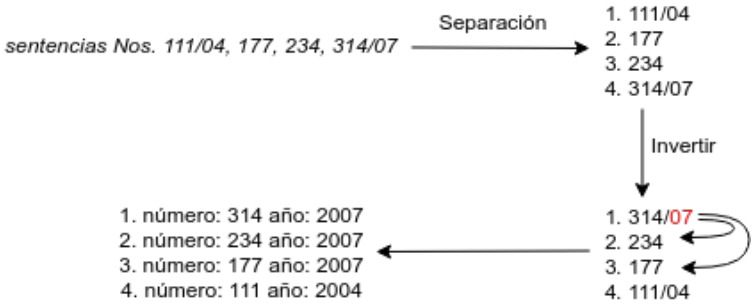

Figura 7.5: Asignación de año en múltiples Sentencias

En el caso en que en la referencia no se reconozca un número o año de Sentencia, el proceso termina. En caso contrario, se procede a la extracción de la sede de la sentencia, la cual puede ser juzgado, tribunal o la Suprema Corte de Justicia. Para esto se aplica una expresión regular formada por 4 grupos:

- Grupo 1: Define un patrón que busca la ocurrencia de palabras que generalmente indican el inicio de una referencia a un tribunal, juzgado o mismo a la Suprema Corte de Justicia como por ejemplo: "tribunal", "juzgado", "suprema", "juez", seguido de una secuencia de palabras y números hasta la aparición de la palabra "turno". Por ejemplo en la referencia VI) de los ejemplos anteriores el segmento de texto que coincidiría con el patrón buscado por este grupo sería " Juez Letrado de Primera Instancia de Treinta y Tres".
- Grupo 2: Define un patrón similar al del Grupo 1 pero sin la restricción de que deba aparecer la palabra "turno".
- Grupo 3: Define patrones para las abreviaciones de Suprema Corte de Justicia por ejemplo "S.C.J".
- Grupo 4: Define patrones de abreviaciones de tribunales, como por ejemplo: "TAC" para el "Tribunal de Apelaciones en lo Civil" o "TCA" para "Tribunal en lo Contencioso y Administrativo".

A su vez, dentro de la información de la sede, se puede incluir la localidad de la misma. Resulta de interés su reconocimiento para luego poder homogeneizar las referencias a sentencias. Para el reconocimiento de ciudades se partió de una lista de las ciudades más importantes de cada departamento del país [36], la cual fue utilizada en distintas formas, dentro de las cuales se puede destacar la utilización de algoritmos de distancia entre textos proporcionados por la librería "textdistance" de Python. Una de las alternativas tenidas en cuenta utilizando esta librería consistió en tokenizar el segmento de texto de la sede de la sentencia, y luego aplicar algoritmos de distancias de palabras como "Damerau-Levenshtein" definiendo como aceptable hasta una diferencia de uno. Esto resulta muy útil para trabajos como el presente, debido a que en la escritura es

muy probable que ocurran transposiciones de caracteres no deseadas. Sin embargo, la opción de tokenizado fue descartada debido a que existen localidades con más de un token, como por ejemplo "Ciudad de la Costa". Otra alternativa tenida en cuenta fue unir todas las palabras de la sede y de las ciudades, para luego aplicar el algoritmo de distancia con el mismo nivel de aceptación. Esta alternativa solucionaba problemas de tipeo como por ejemplo "Monte video", pero no resultó del todo correcta debido a que ciudades con nombres de longitud pequeña como por ejemplo "Melo" poseen palabras con distancia 1 que pueden resultar de unir dos palabras distintas. Finalmente, se optó por no utilizar el algoritmo de distancia pero sí unir los tokens del texto para realizar una comparación exacta con los nombres de las ciudades. Un ejemplo del método utilizado se puede ver con la referencia VI), la cual posee el nombre "Treinta y Tres". La iteración que reconoce a dicha ciudad es aquella en la cual se realiza una búsqueda del string "treintaytres" en el texto "juezletradodeprimerainstanciade**treintaytres**".

Artículos

Las propiedades de interés de las referencias a Artículos consisten en aquella información necesaria para ubicar al segmento de texto del Artículo al cual se hace referencia. Dentro de esta información se destacan su número y su fuente. Además, resulta de interés reconocer aquella información que especifica la ubicación del segmento de texto dentro de la redacción del Artículo al cual se hace referencia. Dentro de esta información se reconocen numerales, incisos y también literales. Cabe destacar que en el caso de este tipo de referencias, se tomaron como válidas únicamente a aquellas que contaran con su fuente de manera explícita, debido a que sin ella es imposible determinar a qué Artículo se está haciendo referencia. A su vez, se asumió que cada referencia extraída por el modelo contaba únicamente con una fuente, la cual podía ser una Ley, Decreto o Cuerpo Normativo (por ejemplo CGP, o CC).

Al igual que en el caso de los Decretos, las referencias extraídas y clasificadas con este tipo pueden contener referencias a más de un Artículo, por ejemplo: "artículos 295, 296 nrales. 5o. y 6o., 297 a 299, 306, 310, 379, 380, 335 del Código de Comercio". Sin embargo, teniendo en cuenta el criterio mencionado en el párrafo anterior, todos los artículos existentes en la referencia tienen como fuente a la misma entidad. Es por esto, que al iniciar el proceso de extracción de propiedades lo primero que se realiza es una búsqueda de los Cuerpos Normativos sobre la referencia completa, para luego poder relacionar a cada artículo con su fuente. Para esto, se utiliza una lista de Cuerpos Normativos generada en el proceso de anotación, la cual contiene diferentes formatos de escritura para cada uno. En el ejemplo presentado, el primer paso del proceso reconoce al Cuerpo Normativo "Código de Comercio" como la fuente.

El procesamiento de los Artículos se basó en la idea principal de que la referencia se puede leer de izquierda a derecha, encontrando las propiedades en secuencia, es decir que si aparece un derivado de la palabra "artículo" es altamente probable que lo próximo que aparezca sea su número. Lo mismo se cumple para los numerales, incisos y literales. También se sabe que si en la secuencia aparece el token "ley" o similar, es altamente probable que los tokens siguientes se correspondan a la información de la misma. Es por esto que el siguiente paso consistió en tokenizar la referencia y procesar los tokens resultantes de principio a fin ejecutando diferentes acciones según corresponda.

Se definieron distintas condiciones evaluadas mediante expresiones regulares que debe cumplir un token para que se realice una acción, estas son:

I) Si el token es un derivado de la palabra “articulo”. Por ejemplo “art.”, “a.”, “articulos”.

II) Si el token es un derivado de la palabra “numeral”. Por ejemplo “num.”, “nral”, “numerales”.

III) Si el token es un derivado de la palabra “inciso”. Por ejemplo “inc”, “incs.”, “incisos”.

IV) Si el token es un derivado de la palabra “literal”. Por ejemplo “lit”, “literales”.

V) Si el token es la palabra “ley”.

VI) Si el token es un derivado de la palabra “decreto”. Por ejemplo “dec”, “dec-ley”, “d-ley”.

VII) Si el token representa un separador de números para indicar un rango. Por ejemplo “a”, “-”, “/”.

VIII) Si el token representa números. Por ejemplo “15780”, “15.780”, “1ero” o “200.1” lo cual corresponde al articulo 200 numeral 1.

IX) Si el token representa una letra.

X) Si el token es el signo de puntuación “;”.

XI) Si el token es el signo de puntuación “,”.

Para poder relacionar los tokens se mantiene un contexto a lo largo de la iteración el cual contiene información necesaria para la toma de decisiones, como por ejemplo qué clase de propiedad se está formando. Por lo tanto, si el contexto indica que se está formando un inciso, el/los números siguientes se corresponderán a incisos. En el contexto también se almacenó una lista de pares (clave,valor) para representar a las propiedades reconocidas hasta el momento, la cual en algunos casos fue de utilidad para desambiguar a qué entidad correspondía determinado número.

Cuando se procesa un token que cumple con alguna de las condiciones I) a V) se almacena en el contexto el tipo del token. Es decir, que si llega un token correspondiente a un numeral, el contexto almacenará la información de que el tipo que se está procesando es “numeral”. Si un token cumple la condición VII), la cual consiste en un separador de rango numérico precedido por un número, se almacena en el contexto dicha información para luego poder construir los números restantes. Luego, las condiciones VIII) y IX) se encargan de reconocer los números o letras correspondientes a la propiedad que se está extrayendo. En particular, la condición IX) reconoce a la letra correspondiente al literal, para luego almacenar en la lista de propiedades el valor de la letra acompañado de la clave “Literal”.

Por otro lado, la condición VIII) no sólo reconoce números que se pueden corresponder con el tipo de propiedad sino que también reconoce el caso particular en el cual en el mismo token coexisten el número del artículo y el numeral asociado a éste. En esta condición se debió tener en cuenta si se estaba procesando un rango numérico, debido a que en dicho caso se debía utilizar el último número reconocido hasta el momento para insertar en la lista un elemento por número dentro del rango correspondiente. Este elemento consta de un par con clave igual al tipo de propiedad reconocido anteriormente en la secuencia y valor igual al

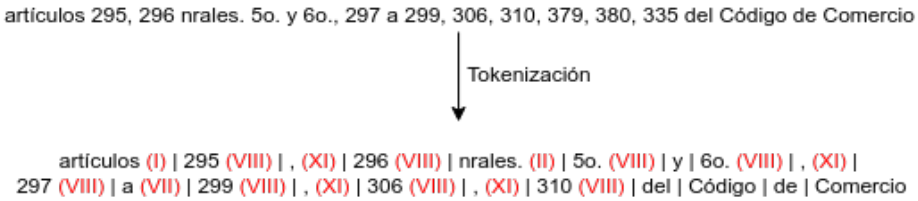

Figura 7.6: Impacto de condiciones sobre tokens de una referencia

número dentro del rango. La condición VI) difiere de las condiciones I) a V) en que no se marca el tipo de token (en este caso correspondiente a un decreto) sino que en dicho paso se procesa el resto del texto de la referencia para obtener la información correspondiente al decreto. Es decir, si se procesa un token derivado de la palabra "decreto" se utiliza lo mencionado en 7.3.1. Para definir el segmento de texto de la referencia de Articulo candidato a ser un decreto, se debió tener en cuenta que éste puede preceder al número del artículo, como pasa en "Decreto N345/983, artículo 294". Por lo tanto, si en lo que resta del texto de la referencia aparece un token que cumpla la condición I), el texto candidato sobre el cual se realiza la búsqueda de la información del decreto es el que va desde el token actual (reconocido como comienzo de un decreto) hasta el token que cumpla dicha condición. En caso contrario se realiza la búsqueda sobre el segmento de texto que va desde el token actual hasta el final de la referencia al Articulo. Una vez que se obtiene la información del decreto, esta es almacenada en la lista de propiedades junto a la clave "Decreto".

En el caso en el que el token cumpla la condición X), el símbolo ";" es agregado a la lista y se almacena en el contexto el tipo "Articulo" debido a que en las referencias obtenidas, dicho símbolo corresponde al fin de un Articulo y el comienzo de otro. Finalmente, si el token es el símbolo "," este es agregado a la lista ya que en el procesamiento posterior es de utilidad para desambiguar escenarios en los cuales no se distingue el límite entre un Artículo y otro.

7.3.2. Transformación de la información

El siguiente paso luego de la extracción de propiedades de los distintos objetos legales se basó en utilizar dicha información para lograr una representación única de cada uno. Es decir, transformar las propiedades de manera de homogeneizar dicha información y lograr que diferentes maneras de expresar lo mismo converjan en una sola. Esto se realizó para generar un grafo interconectado y relacionado con nodos en común entre sentencias. Esto es necesario ya que en diferentes sentencias (o algunas veces hasta en una misma sentencia) se expresa una referencia a un objeto legal de diversas maneras, como por ejemplo "artículo 137 del Código General del Proceso" y "art. 137 C.G.P.". Una consecuencia de extraer esta información y no transformarla a un formato común es que al momento de generar el grafo, para una misma referencia se estarían creando más de un nodo, lo que llevaría a obtener un grafo más desconectado y con nodos redundantes.

Para llevar a cabo este proceso se definieron varios formatos según el tipo de objeto legal a procesar. En el caso de los Doctrinos, lo único que se realizó fue un mapeo entre las ocurrencias de los distintos nombres extraídos en la sección 6.1 y sus nombres completos. Para el caso de decreto, como se mencionó anteriormente, era necesario contar con el número y el año del mismo. Por lo general, los años asociados a los decretos

se suelen expresar con números de entre dos y cuatro cifras, por lo tanto, se tomó la decisión de expresar el año del decreto en el formato de cuatro cifras, agregando cifras delante en caso de ser necesario, ya que es el que brinda mayor cantidad de información. Para saber qué cifra agregar delante en caso de contener solamente dos, se tomó en cuenta la primera cifra de las dos, ya que en caso de ser un 0 o un 1, es altamente probable que se esté haciendo referencia a un año posterior al 2000, mientras que en cualquier otro caso se entiende que se estaría haciendo referencia a un año anterior al 2000 pero posterior al 1900. Por otro lado, el número del decreto se podía encontrar separado en dos números (15.870) o solamente un número (15870), por lo que se decidió en ambos casos mantenerlo como un solo número.

En cuanto a las leyes, el formato utilizado siempre fue de hasta 5 números, pudiendo estar separadas las unidades de mil con un punto o no. Es por esto que se tomó la decisión de mantener un formato de 5 números sin separación, por el mismo motivo que en decreto. El procedimiento a seguir para el año de las fichas fue el mismo que para decreto, pero en el número se decidió mantenerlo en formato IUE (separando en dos números).

Para las sentencias, el proceso se complejizó debido a que está conformada por cuatro propiedades, donde la sede se expresa como texto mientas que turno puede estar expresado con un número o con texto. En cuanto al número de la sentencia, este siempre se presenta como un número de hasta 4 cifras, por lo que se lo almacenó como se presentaba y para el año se utilizó el mismo procedimiento que para ficha y decreto. En el caso del turno se intenta llegar a una única representación numérica partiendo del texto reconocido por la expresión regular. El objetivo es que por ejemplo, tanto el texto “Cuarto Turno” como “4to T” deriven simplemente en el número “4”. Para lograr esto, se desarrolló un mecanismo para llevar a los turnos expresados como palabras a su número correspondiente. Fue necesario realizar un análisis manual de las sentencias de la BJN para determinar los valores de turnos que usualmente son manejados. Luego de esto, se definieron las correspondencias entre los números del 1 al 9 y sus representaciones en palabras. Con esto, se abarcan los casos en los que el turno es menor a 10, pero para los casos contrarios se tuvieron en cuenta los prefijos utilizados comúnmente para escribir estos números. Por ejemplo, “Decimocuarto Turno” o “Vigésimotercer Turno” hacen referencia a los turnos 14 y 23 respectivamente. Por lo tanto, en los casos en los que se reconocieran prefijos como “decimo”, “vigésimo” o “trigésimo”, el número resultante comenzaría con las cifras 1, 2 o 3 respectivamente. Luego, el número es completado teniendo en cuenta la correspondencia para números del 1 al 9 mencionada anteriormente. La sede se almacenó teniendo en cuenta, en el caso en el que fuera un tribunal o juzgado, su tipo (por ejemplo de apelaciones o de lo contencioso y administrativo), su área (por ejemplo civil, penal, de familia, del trabajo) y en caso de estar explícita en la referencia, su procedencia (por ejemplo de Montevideo, de Rivera). En caso de no haberse detectado el tipo o el área, la propiedad era descartada ya que no era de utilidad agregar una propiedad con información incompleta.

Finalmente, para los artículos, el proceso también fue más complejo debido a su composición. Esto es porque los artículos pueden contener además de su número y fuente, numerales, incisos o literales que especifiquen al mismo. En el caso de la fuente, se realizó un mapeo entre los distintos formatos de fuente existentes en el archivo utilizado para detectar las mismas, y una representación única de ellas. En cuanto al número, numeral e inciso, el formato en que aparecían siempre era numérico, por lo tanto se decidió mantenerlos de esa forma. De igual manera se trató a los literales, los cuales siempre se encuentran descriptos con letras.

7.3.3. Carga de la información

Como última parte del proceso de creación del grafo de referencias, se encuentra la creación de los nodos y las relaciones que componen el mismo. Como se expresó en la sección 7.3, la misma se realiza mediante consultas en el lenguaje Cypher. Como muestra la figura 7.2, para cada sentencia de la BJN se crea un nodo al cual se lo denomina como nodo raíz. Luego, mediante los procesos descriptos en las secciones anteriores, se obtienen las propiedades normalizadas de cada referencia existente en el texto de la sentencia raíz, las cuales son utilizadas para generar las consultas y almacenar la información en el grafo de referencias. Por lo tanto cada sentencia raíz tendrá tantas aristas salientes como referencias diferentes existentes en su redacción y tantas entrantes como referencias existentes desde otras sentencias hacia ella.

Para la creación de los nodos y relaciones se utilizaron las cláusulas MERGE y MATCH. La primera es una variante de la cláusula CREATE, con la diferencia de que se crea el nodo o la relación únicamente en el caso en el que no exista un nodo con las propiedades indicadas o una relación con nodos y propiedades que se ajusten a los indicados. Por otro lado, la cláusula MATCH se utilizó como complemento de la cláusula MERGE para evitar múltiples relaciones erróneas. Este problema se dio debido a la forma de procesar las consultas que utilizan Cypher y Neo4j, ya que como se mencionó anteriormente se basan en patrones. Por lo tanto, en casos en los que se especifican por ejemplo dos propiedades, se tendrán en cuenta a la hora de la operación todos los nodos que tengan los valores indicados para dichas dos propiedades, incluyendo así nodos con más de dos propiedades y quizás no deseados.

El proceso por lo tanto se basó en, una vez obtenida la información de la referencia, y contando con la información de la sentencia raíz, crear una única consulta que mediante la utilización de las operaciones mencionadas anteriormente creara los nodos y las relaciones correspondientes entre ellos. Esto fue posible debido a que Cypher permite reutilizar las variables definidas en cláusulas anteriores. Como se puede ver en 7.7, el cual muestra la creación de un nodo de tipo Ficha referenciado por la Sentencia de la BJN 71/2004. Dicha sentencia se define con la variable “s”, la cual es reutilizada para definir la relación con el nodo “x”.

```
MERGE (s:Sentencia {numero: ‘71’, año: ‘2004’, tribunal_juzgado: ‘Tribunal de Apelaciones
en lo Civil’, turno: 2, ficha: ‘369-954/2004’})
MERGE (x:Ficha {numero: ‘369-954’, año: ‘2004’})
MERGE (s)-[r:Expediente]->(x)
```

Figura 7.7: Creación nodo Ficha referenciado por Sentencia

Para las referencias a nodos de tipo Ficha, Decreto, Ley y Doctrino, el proceso consiste en anidar tres cláusulas MERGE de la misma forma que en 7.7. El primer MERGE tiene dos funciones, la primera vez que se ejecuta se encarga de crear el nodo raíz como se ve en la primera figura de 7.8 mientras que en las siguientes ocurrencias se encarga de cargar en la variable “s” el nodo creado anteriormente. El segundo MERGE se corresponde con la creación del nodo correspondiente al objeto legal que ha sido referenciado desde la sentencia raíz. Como se mencionó anteriormente, dicha cláusula creará el nodo únicamente en el caso en el que ningún nodo existente cumpla con el patrón definido en el MERGE. De todas formas, el nodo asociado al objeto legal se carga en la variable “x”. Por último, se crea la relación entre el nodo cargado en

la variable “s” y el cargado en la variable “x”. Notar que la relación es unidireccional y toma sentido desde el nodo raíz hacia el nodo del objeto legal referido. En la figura 7.8 se puede observar el resultado del proceso completo de carga de la información de una referencia desde una Sentencia hacia una Ficha. También se puede ver, que las propiedades de los objetos legales forman parte del nodo.

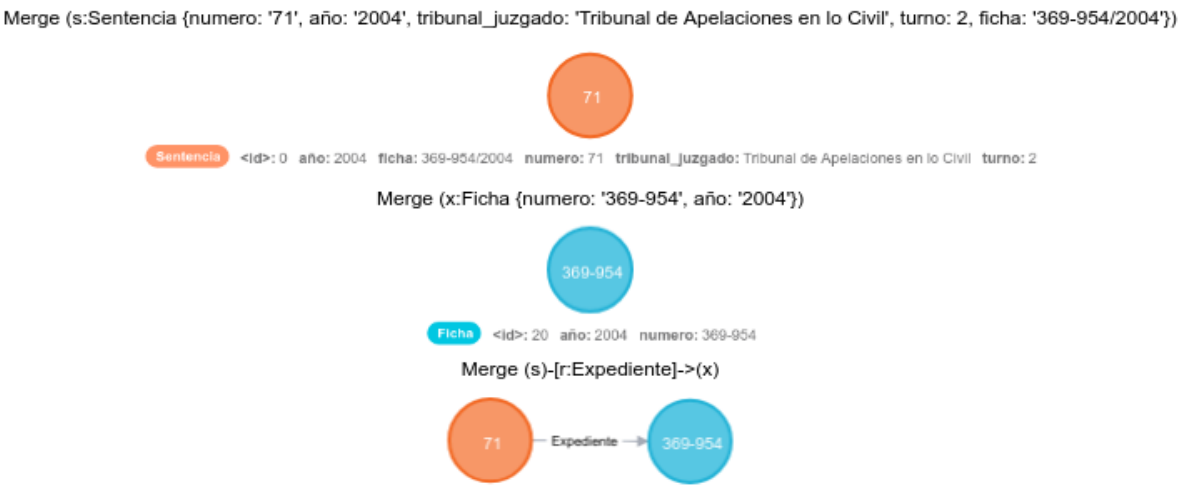

Figura 7.8: Creación referencia Sentencia a Ficha

El proceso de carga de nodos de tipo Artículo difiere a lo expresado en el párrafo anterior debido a que no sólo es necesario crear un nodo asociado a dicho artículo, sino que también es necesario crear un nodo asociado a la fuente del mismo. Esto se realiza con el objetivo de que exista un único nodo por fuente con tantas aristas entrantes como referencias a artículos existentes con dicha fuente. Por lo tanto la consulta final correspondiente a la carga de una referencia a Artículo no está compuesta únicamente por tres cláusulas MERGE, sino que se agregan dos cláusulas más. Como se puede ver en 7.9, una de ellas se corresponde con la creación o simplemente la búsqueda del nodo correspondiente a la fuente del Artículo (“MERGE (l:CuerpoNormativo {nombre: ‘Código General del Proceso’})”), mientras que la restante se corresponde con la creación de la relación de tipo “Fuente” entre el nodo Artículo y el nodo fuente. Otro punto a destacar, es que en este caso no todas las propiedades del objeto legal forman parte del nodo como atributos, sino que las únicas propiedades que forman parte del nodo de un Artículo son su número y su fuente. Las propiedades restantes como son los numerales, incisos y literales forman parte de la relación entre el nodo raíz y el nodo del Artículo. De esta forma, se logra mantener solamente un nodo para cada Artículo, diferenciando las referencias hacia el mismo con varias relaciones entre nodos raíces y el nodo del objeto legal.

```
MERGE (s:Sentencia {numero: ‘71’, año: ‘2004’, tribunal_juzgado: ‘Tribunal de Apelaciones
en lo Civil’, turno: 2, ficha: ‘369-954/2004’})
MERGE (x:Articulo {numero: ‘200’, fuente: ‘Código General del Proceso’})
MERGE (l:CuerpoNormativo {nombre: ‘Código General del Proceso’})
MERGE (x)-[f:Fuente]->(l)
MERGE (s)-[r:Referencia {numeral: ‘1,2’}]->(x)
```

Figura 7.9: Creación nodo Artículo referenciado por Sentencia

Por último, el proceso de carga de nodos de tipo Sentencia también posee una particularidad en comparación con el de los objetos de tipo Ficha, Decreto, Ley y Doctrino. Esta particularidad se da debido a que se consideraron como válidas sentencias que solamente presentan número y año, por lo que se daba la situación en que una sentencia estaba representada por más de un nodo con algunas propiedades en común. Esto, como se mencionó anteriormente generaba un problema a la hora de utilizar únicamente cláusulas MERGE, ya que se creaba una relación entre cada nodo Sentencia que tuviera al menos las propiedades indicadas, con el nodo correspondiente al objeto referido. Es por esto que la consulta de este tipo de objeto posee, además de las cláusulas MERGE, una cláusula MATCH la cual es utilizada para obtener únicamente el nodo que posea exactamente las propiedades indicadas en la consulta, lo cual permite que se cree una única relación.

```
MERGE (t:Sentencia {numero: '71', año: '2004', tribunal_juzgado: 'Juzgado Letrado de
Primera Instancia', turno: '2'}) WITH t
MATCH (x:Sentencia {numero: '71', año: '2004', tribunal_juzgado: 'Juzgado Letrado de
Primera Instancia', turno: '2'}) WHERE size(keys(x)) = 4
MERGE (s:Sentencia {numero: '68', año: '2007', ficha: '29-63/2006'})
MERGE (s)-[r:Referencia]->(x)
```

Figura 7.10: Creación nodo Sentencia referenciado por otra Sentencia

Como se puede ver en 7.10, se mantiene el primer MERGE con el objetivo de crear el nodo en caso de que no exista coincidencia. Sin embargo, la variable asignada a dicho nodo no es utilizada ya que en ese caso se estaría cayendo en el problema de múltiples relaciones descripto anteriormente. Luego, la cláusula MATCH restringe la carga de la variable "x" a sólo un nodo, el creado anteriormente o aquel ya existente que cumpla exactamente con las propiedades descriptas.

7.3.4. Ejemplo del proceso

En esta sección se presenta un ejemplo completo de procesamiento de una referencia perteneciente a una sentencia de la BJN. La sentencia de segunda instancia tenida en cuenta en este ejemplo es la número 341 del año 2008 de la Suprema Corte de Justicia. En este caso se ilustrará el proceso de análisis de una referencia a un Artículo del Reglamento Nacional de Circulación Vial. Partiendo del segmento de texto reconocido por el modelo descripto en la sección 5.3 el cual es "artículo 11.4 literal b) del Reglamento Nacional de Circulación Vial", y contando con información de la sentencia como su número, año, sede y ficha, se comienza el proceso de extracción de propiedades.

En este ejemplo, el nodo raíz ya está creado debido a que éste se genera al momento de extraer la información de los metadatos de la sentencia y relacionarla con el nodo correspondiente a su expediente. Como se mencionó a lo largo de las secciones anteriores, el proceso comienza en este caso con la búsqueda del nodo raíz correspondiente a la sentencia analizada. Para esto se utiliza la cláusula MERGE colocando como atributos del nodo a aquellas propiedades extraídas de los metadatos de la sentencia. El atributo ficha se utiliza, como se mencionó anteriormente, para diferenciar a la sentencia raíz de otras sentencias con el mismo número y año en los casos en que no se posee información de la sede o el turno. Por lo tanto, se comienza el proceso de construcción de la consulta en lenguaje Cypher colocando el MERGE correspondiente al nodo

raíz. Luego, iterando en las referencias extraídas por el modelo, se llega a la referencia de Artículo utilizada en este ejemplo. El texto de la misma es enviado al módulo de extracción de propiedades.

Lo primero que se realiza es la búsqueda de una posible fuente. En este caso dicho proceso logra la recuperación de la fuente "Reglamento Nacional de Circulación Vial" y la almacena para luego relacionarla con el Artículo. Luego, el texto pasa por un proceso de normalización en el cual se eliminan acentos y símbolos como por ejemplo los paréntesis. Como siguiente paso, se tokeniza el texto de la referencia y se comienza a iterar sobre los tokens.

Al comienzo del proceso de iteración el contexto contiene el valor "Artículo" como objeto que se está analizando, mientras que la lista de propiedades comienza con el elemento (Fuente, "Reglamento Nacional de Circulación Vial") correspondiente a la fuente encontrada en el primer paso. El primer token procesado es "artículo", lo cual no genera ningún cambio en el contexto. Al continuar se procesa el token "11.4". En este caso se reconoce que se trata de un número, pero debido a que el contexto almacena la información de que se está buscando un Artículo, dicho número corresponde al del objeto legal y además, el número luego del "." se corresponde con un numeral del Artículo. Por lo tanto, en dicha iteración se agregan dos elementos a la lista de propiedades, uno marcando al artículo número 11 y otro marcando al numeral 4. En este caso el contexto sigue manteniendo el valor "Artículo" debido a que si el numeral está junto al número del artículo, se puede afirmar que en caso de que ocurra un número luego, este no será de un numeral ya que esta notación se utiliza en casos en los que se referencia a un único numeral.

La siguiente iteración toma al token "literal" y actualiza el contexto indicando que lo próximo a buscar es la letra de un literal. Luego se procesa el token "b", el cual cumple con la condición IX) detallada previamente en esta sección. Debido a que el contexto mantiene el valor "Literal" como referencia al elemento buscado, se agrega un elemento a la lista de propiedades con el valor ("Literal", "b"). Luego de esto, se procesa el resto de los tokens los cuales no cumplen con ninguna de las once condiciones utilizadas para ejecutar acciones sobre el contexto. De esta forma, la lista de propiedades no sufre más modificaciones y culmina el proceso de extracción de propiedades, dando paso a la utilización de las mismas para generar información homogénea que caracterice al Artículo que está siendo referido.

El siguiente paso es analizar la secuencia de elementos encontrados en el proceso anterior, e ir ordenando y formando las propiedades completas a utilizar como atributos del nodo. Para esto se tuvo en cuenta el tipo de elemento, el cual depende de las condiciones que hayan cumplido los tokens de la referencia. La lista resultante de procesar el Artículo del ejemplo es la siguiente:

- (Fuente, Reglamento Nacional de Circulación Vial)
- (Articulo, 11)
- (Numeral, 4)
- (Literal, b)

Para construir los atributos del nodo se comienza definiendo un objeto con cinco propiedades: número, numeral, inciso, literal y fuente. Luego se comienzan a procesar los elementos de la lista en orden, tomando así en primera instancia al elemento correspondiente a la fuente. Dependiendo del tipo de la fuente (Decreto, Ley o Cuerpo Normativo) es cómo se transforma a la misma. En este caso no se le realiza ninguna modificación ya que se obtuvo su nombre completo a partir de la lista de Cuerpos Normativos conocida.

Debido a que una referencia puede contener a varios Artículos, es necesario realizar un control para saber cuándo se culmina con la información de un Artículo y se comienza a procesar la de otro. En este caso no sucede, pero cabe destacar que en el proceso, si se analiza un elemento de tipo "Articulo", antes de cargar el campo número del objeto con el valor del elemento, se consulta si dicho campo no está vacío. Si éste no lo estuviera, significa que se culminó con la construcción de los datos de un Artículo, y a partir del elemento actual se comienza a procesar un nuevo Artículo (proveniente de la misma referencia).

En el presente ejemplo, se procesa el elemento (Articulo, 11) asignándole al objeto el valor 11 en el campo número. Siguiendo con los elementos, llega el turno de (Numeral, 4), el cual es agregado al conjunto de numerales del objeto. Luego, con el elemento (Literal, b) se carga la variable literal del objeto con el valor "b". Esto se realiza sin restricciones debido a que se sabe que estos elementos se corresponden con el Artículo actual y no refieren a un Artículo que comienza. El fin y comienzo de un artículo se puede dar cuando existen elementos resultantes de procesar el token "," o ";".

Al llegar al fin de la lista, se envía el objeto formado con la información del Artículo junto con la información del nodo raíz hacia el módulo de carga de datos, el cual simplemente utiliza la información recibida para crear la consulta de carga correspondiente. De esta forma se ejecuta la consulta siguiendo un proceso similar al descripto por la figura 7.8, resultando en el subgrafo representado por la figura 7.11. Como se mencionó previamente, las propiedades tenidas en cuenta para representar a un Artículo aportan en conjunto para la identificación exacta del extracto de texto del Artículo al cual se hace referencia. En este caso la ocurrencia tanto del literal b como del numeral 4 significa que el segmento de texto al cual se hace referencia es el correspondiente al literal b del numeral 4 del artículo 11 del Reglamento Nacional de Circulación Vial.

Figura 7.11: Subgrafo ejemplo

7.4. Problemas

Información de sedes

Al comienzo del pre-procesamiento se encontró el problema de que los datos obtenidos en formato CSV, en sus metadatos contenían el número y el año de la sentencia a la cual referían, pero no así su turno ni su sede. Como se mencionó en secciones anteriores, para definir completamente una sentencia, es necesario tener estas dos propiedades. Además, no tenerlas generaba problemas al intentar hacer MERGE ya que generalizaba y tenía en cuenta todas las sentencias con al menos esas propiedades. Por lo tanto, para solucionar este problema se optó por agregar en las propiedades de las sentencias raíz, la ficha o expediente en el que se encontraban, el cual era accesible desde los metadatos, evitando de esta manera confundir las sentencias raíz con otras sentencias referidas dentro de los textos de la BJN.

Sin embargo, avanzado el desarrollo del sistema, se pudo contar con la información de las sedes almacenadas en la BJN. Los datos con los que se contaba al principio, presentaban para cada sentencia el identificador de la sede en la cual se emitió la misma pero no así su descripción. Al obtener esta información, se pudo realizar un mapeo y asignarle la información del turno y sede a las sentencias raíz, quedando estas con cinco propiedades: número, año, ficha, sede y turno. De esta manera, los nodos raíz quedaron plenamente identificados lo cual disminuyó la probabilidad de que el sistema cree relaciones erróneas entre nodos.

Cantidad de elementos

Durante el desarrollo del sistema, se debieron realizar diferentes ajustes debido al comportamiento y al desconocimiento del lenguaje de consulta Cypher. Como se mencionó anteriormente, la cláusula MERGE recupera todos aquellos nodos que cumplan con las propiedades indicadas o más. En el ejemplo mostrado en 7.12 se puede ver la existencia de dos nodos de tipo Sentencia con propiedades en común, su año y su número. Luego, se puede ver el efecto de utilizar únicamente la cláusula MERGE, habiéndose creado, en una única consulta, dos relaciones desde el nodo correspondiente a la Sentencia número 97 a los nodos con número 200. En una primera etapa, este comportamiento llevó a que se generaran un gran número de relaciones incorrectas, acompañado de una demora excesiva en el procesamiento. Los primeros resultados obtenidos contaron con aproximadamente 10 millones de relaciones. Es por esto que, como se mencionó anteriormente, se recurrió a la combinación de la cláusula MATCH con la adición del número de Ficha en los nodos raíz para solucionar el problema mencionado y disminuir la cantidad de relaciones.

Figura 7.12: Múltiples referencias

Visualización de los datos

Otro problema que se tuvo que tratar en este trabajo está relacionado con la visualización de los nodos y relaciones del grafo. Luego de realizar el post procesamiento y la carga de datos, y al ser el grafo resultante de gran tamaño, superando los 250 mil nodos y el millón de relaciones, se hizo muy difícil poder obtener una visualización completa del mismo. Esto es debido a la capacidad de procesamiento gráfico requerida para poder desplegar todos los nodos y relaciones en pantalla. Sin embargo, resulta lógico que el interés principal no es ver el grafo completo sino que su utilidad se basa en poder analizar los comportamientos de diferentes conjuntos de nodos. En las siguientes secciones se plantean algunas posibles consultas para poder obtener información más acotada del grafo de una manera más amigable y entendible para el usuario.

Tiempo de ejecución

Debido a la gran cantidad de datos a procesar, un problema a enfrentar fue el del tiempo de ejecución. La primera ejecución del sistema sobre todas las sentencias de la BJN culminó en un tiempo de 76 horas y 38 minutos. La causa de este problema fue la velocidad a la cual crecía la cantidad de nodos y relaciones del grafo por problemas como el mencionado en la sección 7.4. Como se mencionó en secciones anteriores, el motor de consultas de Cypher realiza una búsqueda por patrones, transformando así una cláusula MATCH en una gran cantidad de comparaciones de nodos. Debido a que en las consultas generadas para la creación de nodos de tipo Sentencia se utiliza esta cláusula, y sumado a la gran cantidad de referencias a objetos de este tipo existentes en la BJN, se estaba comparando a cada nuevo nodo Sentencia con los ya existentes.

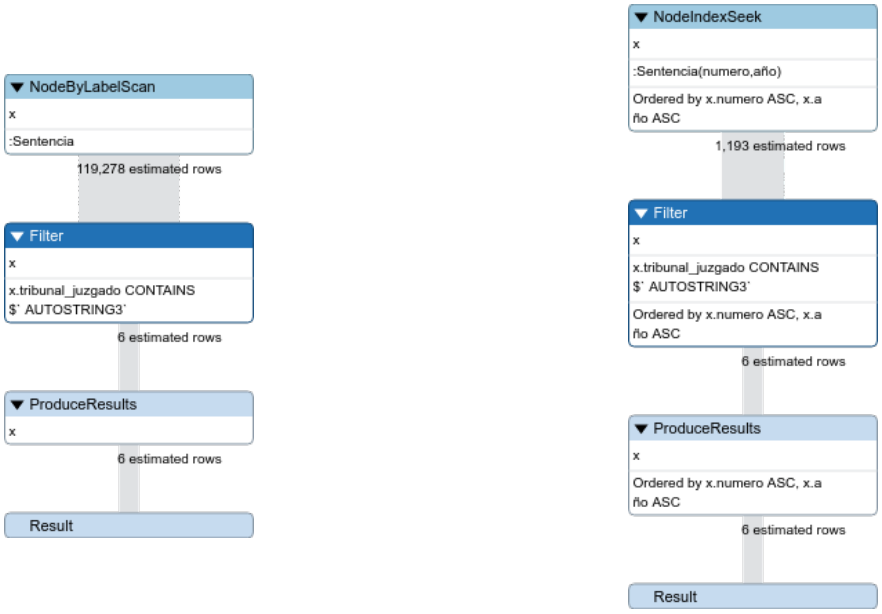

Figura 7.13: Plan de ejecución sin índices vs con índices

Para mitigar este problema se introdujo la utilización de índices por propiedades, aprovechando la flexibilidad que ofrecen Neo4j y Cypher para su creación. Se decidió crear índices para los tipos de nodos Sentencia, Articulo, Ley, Decreto y Ficha, dejando afuera a los Doctrinos y Cuerpos Normativos debido a

que sus cantidades son despreciables. Con la posibilidad de utilizar índices compuestos disponible en Neo4j, se procedió a crearlos para aquellas propiedades que siempre están presentes en cada tipo de nodo. Por ejemplo número y año para los casos de Sentencia, Decreto y Ficha, número y fuente para Artículo y número para Ley. La figura 7.13 muestra, para una consulta de carga de un nodo de tipo Sentencia, una comparación entre los planes de ejecución del fragmento correspondiente a la cláusula MATCH de dicha consulta con índices definidos y sin ellos.

Cuando no se cuenta con índices el plan presentado en la figura 7.13 muestra una búsqueda por tipo de nodo. El término "NodeByLabelScan" visible en la imagen de la izquierda, hace referencia a que se utilizará únicamente un índice por la etiqueta Sentencia. En la imagen también se puede ver que luego de dicha búsqueda el sistema estima que se obtendrá un conjunto de aproximadamente 119278 nodos de tipo Sentencia al cual se le aplicarán los filtros indicados en la consulta. Por otro lado, el plan sobre un grafo en el cual se configuraron los índices mencionados anteriormente muestra la aplicación del operador "NodeIndexSeek", el cual utiliza el índice para nodos de tipo Sentencia sobre las propiedades número y año. Este operador estima que se obtendrá un conjunto de aproximadamente 1193 nodos de tipo Sentencia, el cual es cien veces más chico que el que se obtiene al no utilizar índices. Con el objetivo de comparar los rendimientos de las bases con índices y sin índices se ejecutaron diversas consultas. Una de ellas es la subconsulta MATCH que se muestra en el ejemplo 7.10 correspondiente a la Sentencia 71 del año 2004. Esta consulta conllevó 2ms en el ambiente con índices y 173ms en el ambiente sin índices, comprobando la eficacia de la utilización de los mismos. Luego de ver esta diferencia, se modificó el sistema para crear los índices y se procedió a ejecutarlo nuevamente, culminando dicha ejecución en un tiempo de 21 horas y 12 minutos.

Redacción de las sentencias

Los errores presentes en muchas de las redacciones de las sentencias significaron un gran problema en varias etapas de este trabajo. Particularmente en esta etapa, este factor incrementó la dificultad de extraer correctamente la información correspondiente a las propiedades del objeto legal referido. Un ejemplo de este problema se puede ver en el segmento de texto extraído por el modelo SpaCy "arts. 197, 1981 254 y 257 CGP". En dicha referencia, no se distingue el significado del número 1981, ya que cerca de él no se encuentra ninguna palabra característica de artículo, ley u otro elemento, pero mirando el contexto se puede deducir que no se trata de un número de artículo. Por lo tanto, el sistema procesa incorrectamente este caso seleccionando a dicho número como si fuera un número de artículo. En realidad, en vez de "1981" debería estar escrito "198.1" lo cual hace referencia al numeral 1 del artículo 198.

Otro problema que surge a partir de la forma en la que se redactan las sentencias es el de los nombres de Doctrinos. En muchos casos, las referencias a éstos se realizan mediante parte de su nombre completo o únicamente con su apellido. Por lo tanto, al querer realizar un proceso de extracción automática de los mismos, surge un problema que consiste en que si se deseara realizar la búsqueda utilizando el nombre completo del Doctrino seguramente se encontrarían muy pocas coincidencias, ya que por lo mencionado anteriormente esta no es la forma más común de referenciarlos. Sin embargo, si se utilizaran partes de su nombre completo, por ejemplo únicamente "Gamarra" en el caso de "Jorge Gamarra", se obtendrían muchos falsos positivos afectando luego a la consistencia del grafo. Dicho esto, se entiende que este problema posee una complejidad que lo obliga a ser tratado con más énfasis, escapando del alcance de este trabajo.

7.5. Resultados

En esta sección se presentan los resultados obtenidos luego de la ejecución del sistema sobre la totalidad de sentencias almacenadas en la BJN. Además, se presentan diferentes ejemplos de análisis de datos que se pueden realizar sobre el grafo, aprovechando la posibilidad de visualizar los resultados de manera más intuitiva.

La ejecución del sistema se realizó sobre el equipo descripto al comienzo de este informe, obteniendo un total de 250343 nodos y 1116278 relaciones. En la imagen 7.14 se puede ver la estructura final del grafo de referencias. Esta imagen muestra los diferentes tipos de nodos que se pueden encontrar y qué relaciones existen entre los mismos. El tipo de nodo principal son las Sentencias, las cuales tienen aristas salientes de tipo "Referencia" hacia los demás objetos legales, incluyendo un lazo debido a que pueden existir referencias entre distintas Sentencias. Por otro lado, el esquema muestra que los nodos de tipo Artículo tienen relaciones de tipo "Fuente" con otros nodos, dentro de los cuales se encuentran los de tipo "Cuerpo Normativo", los cuales aunque no forman parte del conjunto de objetos legales considerados para el entrenamiento del modelo, fueron tenidos en cuenta en la etapa de transformación para que el artículo se encuentre mejor definido. Otro detalle a destacar de la estructura del grafo es que existen dos tipos de relaciones entre los nodos de tipo Sentencia y los de tipo Ficha. Una de ellas está etiquetada como "Expediente" y se corresponde a la relación entre una Sentencia raíz (correspondiente a una sentencia de la BJN) y el expediente del caso extraído de los metadatos de la misma, mientras que la otra está etiquetada como "Referencia" y se corresponde con una relación entre una Sentencia raíz y una ficha correspondiente a otro caso.

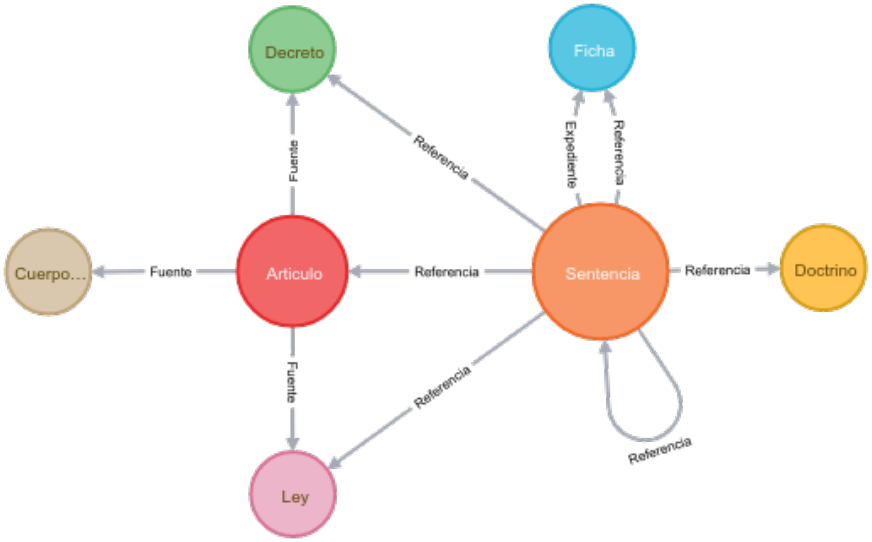

Figura 7.14: Estructura de Grafo

La tabla 7.1 muestra para cada tipo de objeto legal, la cantidad de nodos cargados en el grafo así como la cantidad de relaciones (aristas salientes) desde nodos de tipo Artículo y Sentencia hacia los demás objetos. Cabe destacar que se obviaron las filas correspondientes a los objetos legales restantes debido a que estos, al no referenciar a ningún otro objeto, no poseen aristas salientes. Se puede ver que el tipo de nodo con más

ocurrencias es el de Sentencias, abarcando más de la mitad de los nodos totales. Esto se debe a que dicho objeto legal es el centro de este trabajo, contando con una base de alrededor de 72000 sentencias por lo que mínimo se tendrían esta cantidad de nodos. Esto sumado a la gran cantidad de referencias cruzadas entre sentencias, dentro de las cuales hay algunas que no se encuentran almacenadas en la BJN, provoca que el número de nodos de este tipo sea tan grande. El segundo objeto con más nodos es la Ficha, esto se da por la misma razón mencionada anteriormente ya que por cada Sentencia existe una Ficha correspondiente al expediente del caso, generando como mínimo 72000 nodos de tipo Ficha.

	Artículo	Sentencia	Decreto	Ley	Ficha	C. Normativo	Doctrino
Artículo	0	0	2108	10990	0	5496	0
Sentencia	630565	189103	18947	34024	149414	0	75631
Nodos	18594	137709	12493	2131	79325	17	74

Tabla 7.1: Cantidad de nodos y relaciones salientes

Si bien los Artículos no ocupan la mayor parte de los nodos existentes en el grafo, son los más referidos. La diferencia entre 18594 nodos de Artículos y 630565 relaciones entre Sentencias y Artículos se debe a que, como se mencionó en la sección 7.3.3, las propiedades inciso, literal y numeral que especifican a las referencias a Artículos, se incluyeron como propiedades de la relación, disminuyendo al cantidad de nodos creados. Un ejemplo de consulta que puede ser utilizada con distintos fines es la que se encarga de buscar el nodo más referenciado de un tipo determinado. Esta consulta se puede ejecutar por ejemplo para conocer el artículo más referenciado dentro de las sentencias de la BJN.

```
MATCH (x:Articulo) WITH x,
SIZE(()-[:Referencia]->(x)) AS CantRef
ORDER BY CantRef DESC LIMIT 1
MATCH (a)-[:Referencia]->(x)
RETURN DISTINCT x.numero, x.fuente, CantRef
```

Figura 7.15: Consulta artículo más referenciado

El resultado de esta consulta mostró que el Artículo más referenciado en las sentencias de la BJN es el 200 del Código General del Proceso. Este artículo fue referenciado 24362 veces, lo cual tiene sentido debido a que el mismo refiere a sentencias de segunda instancia, elementos que componen la BJN utilizada para este trabajo. Además, se puede combinar esta consulta con otros filtros como ser el Cuerpo Normativo de los artículos a tener en cuenta, o la sede de las sentencias que refieren a dicho artículo. En la imagen 7.16 se puede ver una variante de la consulta agregando la condición correspondiente a considerar únicamente aquellos artículos del Cuerpo Normativo “Código Penal” referenciados por una sentencia cuya sede pertenece al área de Derecho Penal. Esta consulta retornó como resultado al artículo 86 del Código Penal, con 6106 referencias. Este artículo trata principalmente sobre cuándo puede un Juez revocar u ordenar la ejecución de una pena sobre un penado.

```
MATCH (x:Articulo) WITH x,
SIZE(()-[:Referencia]->(x)) AS CantRef
ORDER BY CantRef DESC
MATCH (a)-[:Referencia]->(x)
WHERE a.tribunal_juzgado CONTAINS 'Penal' AND x.fuente CONTAINS
'Penal'
RETURN DISTINCT x.numero, x.fuente, CantRef LIMIT 1
```

Figura 7.16: Consulta artículo más referenciado del Código Penal

Otra posible consulta podría ser buscar cual o cuales son las sedes que más referencian a cada Doctrino. De esta forma se puede analizar las áreas del derecho en las cuales se manejan los mismos, surgiendo la posibilidad de observar si un Doctrino, a pesar de dedicarse a un área en concreto, también es referenciado desde Sentencias emitidas por sedes competentes con otras áreas.

```
MATCH (s:Sentencia)-[:Referencia]->(d:Doctrino) WITH d,s,
SIZE((s)-[:Referencia]->(d)) AS CantRef
RETURN s.tribunal_juzgado AS `Sede`, d.nombre AS `Doctrino`,
sum(CantRef) AS `Cantidad de
Referencias`
ORDER BY d.nombre, sum(CantRef) DESC
```

Figura 7.17: Consulta referencias a Doctrinos por Sede

En las figuras 7.18a y 7.18b se puede ver alguno de los resultados de la consulta anterior. Se tomaron los casos de los doctrinos Jorge Gamarra y Américo Plá Rodríguez. Se puede ver que las sentencias pertenecientes al Tribunal de Apelaciones en lo Civil son las que más referencian al primer doctrino, con 3351 referencias. Esto tiene sentido debido a que Jorge Gamarra tiene un vínculo más cercano a los temas del Derecho Civil, por ejemplo siendo autor del libro "Tratado de Derecho Civil Uruguayo". Sin embargo, se puede ver que a este doctrino también se lo referencian muchas veces desde sentencias emitidas por tribunales dedicados a temas laborales y de familia. Otro ejemplo de esto se puede ver con Américo Plá Rodríguez, el cual fue un abogado laboralista y es referenciado muchas veces por sentencias de la Suprema Corte de Justicia y del Tribunal de Apelaciones en lo Civil.

"Tribunal de Apelaciones en lo Civil"	"Jorge Gamarra "	3351
"Suprema Corte de Justicia"	"Jorge Gamarra "	1039
"Tribunal de Apelaciones del Trabajo"	"Jorge Gamarra "	625
"Tribunal de Apelaciones de Familia"	"Jorge Gamarra "	177

(a) Gamarra

"Tribunal de Apelaciones del Trabajo"	"Américo Pla Rodriguez "	3861
"Suprema Corte de Justicia"	"Américo Pla Rodriguez "	421
"Tribunal de Apelaciones en lo Civil"	"Américo Pla Rodriguez "	180

(b) Plá Rodriguez

Figura 7.18: Cantidad de referencias por Doctrino

7.6. Evaluación

Luego de mencionar las distintas etapas del sistema, y sumado a los diferentes problemas enfrentados en el proceso, surge la necesidad de evaluar dicho sistema para obtener un panorama del grado de correctitud del mismo. La idea principal de este proceso de evaluación consiste en ejecutar el sistema mencionado en conjunto con el modelo SpaCy y sin él, para de esta forma evaluar individualmente al sistema de procesamiento y carga de referencias al grafo, y también obtener una evaluación total del sistema completo. Para las dos evaluaciones se utilizó el mismo conjunto de evaluación compuesto de 200 sentencias empleado en las evaluaciones presentadas en secciones anteriores. Para cada ejemplo procesado, el objetivo consiste en determinar si la salida del sistema es la esperada o no.

- **Descartada incorrecta:** `artículo 6 de la ley de Unión Concubinaria`
- **Descartada correcta:** `artículo 6`

Figura 7.19: Clasificación como descartado incorrectamente y descartado correctamente

Para que el proceso de evaluación fuese más fluido se agregó al sistema la posibilidad de que el usuario pueda indicar si la consulta generada es correcta para la referencia analizada. Como se mencionó en la sección 7.3, algunas referencias fueron descartadas por el sistema debido a que carecían de información necesaria para identificar al objeto legal referenciado, por ejemplo un Artículo sin fuente como se ve en el ejemplo de 7.19. Estas referencias fueron tenidas en cuenta en el proceso de evaluación, permitiendo al usuario indicar si fueron correctamente descartadas o incorrectamente descartadas. Por lo tanto se agregaron cuatro opciones de clasificación:

- **Correcto:** para indicar que el procesamiento fue correcto, para los casos en que el sistema procesó a la referencia y generó la consulta correcta que luego daría de alta los nodos y las relaciones en el grafo.
- **Incorrecto:** para indicar que el procesamiento no fue correcto, para los casos en los que la consulta generada presenta errores, es decir, que no refleja correctamente toda la información contenida en la referencia.
- **Descartada correctamente:** para indicar que la referencia no contenía toda la información necesaria para ser procesada y cargada al grafo, por lo que fue correctamente descartada por el sistema. Un ejemplo se puede ver en la segunda referencia de 7.19.
- **Descartada incorrectamente:** para indicar que a pesar de que la referencia contenía toda la información necesaria para ser procesada, el sistema la descartó. Un ejemplo se puede ver en la primera referencia de 7.19.

Por lo tanto, como ejemplos negativos se podrían tener aquellas referencias para las cuales se generó erróneamente la consulta de carga al grafo, y aquellas erróneamente descartadas por el sistema, mientras que como ejemplos positivos se cuentan únicamente aquellas referencias procesadas y generadas correctamente por el mismo. Cabe destacar que aquellas referencias correctamente descartadas no se suman en ninguno de los dos conjuntos.

7.6.1. Evaluación del Sistema de carga al Grafo

En esta sección se detallan los resultados de la evaluación realizada sobre el sistema de procesamiento y carga de referencias a la base de datos de grafos. La tabla 7.2 muestra los resultados del proceso realizado sobre el conjunto de evaluación. Para cumplir con esta tarea, en vez de utilizar la salida del modelo SpaCy como entrada, se pasó a utilizar las anotaciones presentes en el conjunto de evaluación, para de esta forma dejar de lado el posible error cometido por la herramienta de NER. En la tabla se puede ver cantidad total de referencias, la cantidad de referencias procesadas y cargadas correctamente, la cantidad de referencias mal descartadas, la cantidad de referencias procesadas de forma incorrecta y aquellas descartadas correctamente.

En la tabla 7.2 no hay datos sobre las referencias a Doctrinos debido a que estas no fueron evaluadas con este proceso. La razón de esto es que las referencias a este tipo de objetos se resolvieron mediante la búsqueda de los nombres obtenidos en la sección 6.1, por lo que no se requiere de ningún procesamiento posterior.

	Correcto	Incorrecto	Descartado Correcto	Descartado Incorrecto	**Total**
Artículo	1232	67	196	68	1563
Sentencia	573	117	37	9	736
Decreto	33	4	61	13	111
Ley	478	0	0	3	481
Ficha	242	0	0	0	242
Total	2558	188	294	93	3133
Porcentaje	81.65 %	6.00 %	9.38 %	2.97 %	100 %

Tabla 7.2: Cantidad de referencias en evaluación individual

El resultado obtenido contando las referencias descartadas incorrectamente como casos incorrectos alcanza el 90.10 % de acierto, pero si se toman en cuenta únicamente las correctas y las incorrectas, el resultado total asciende a un 93.15 %. Analizando individualmente los casos según el tipo de referencia, se puede ver que para los tipos Ficha y Ley no se registraron referencias procesadas incorrectamente. Esto se debe a que estos objetos poseen a lo sumo dos propiedades entre año y número, por lo que si se reconoció a estos elementos en la referencia, es muy probable que la misma se procese correctamente. Otro objeto legal para el cual se marcaron pocas referencias procesadas incorrectamente fue el Decreto, con 4 referencias incorrectas sobre un total de 111. Uno de estos ejemplos se dio con la anotación "decreto N 2416, del 1.11.2007", la cual contiene el símbolo "." para separar los términos de la fecha. El sistema reconoció como número del artículo al 111 debido a que dentro de las regla creadas para reconocer el número, una de ellas permite el uso del punto dentro de los números y además no se reconoce a este símbolo dentro de las fechas.

Otro punto a destacar es el gran porcentaje (54.95 %) de referencias a Decretos descartadas correctamente. Esto se dio debido a la gran cantidad de ocurrencias de referencias de este tipo que no contaban con el año, propiedad necesaria tenida en cuenta por este sistema a la hora de crear los nodos correspondientes. Por otro lado, se puede ver que para este objeto legal se clasificaron muchas referencias como descartadas

incorrectamente, las cuales se corresponden a referencias que incluyen texto descriptivo del decreto, lo cual no es soportado por el sistema. Un ejemplo de referencia descartada incorrectamente es "decreto de la Junta Departamental de Rocha No. 2/06". Este error se debe a que al haber utilizado reglas manuales para la extracción de propiedades de las referencias, es difícil distinguir entre un Decreto completo y un segmento de texto que contenga una variante de la palabra decreto seguido de otras palabras y números.

En cuanto a los Artículos, se obtuvo un 78.82 % del total de referencias procesado correctamente, mientras que si se quitan las descartadas correctamente se llega a un 90.12 %. El número de referencias descartadas correctamente se debe a que existen muchos casos en los que se refiere completamente a un artículo y posteriormente únicamente se lo referencia mediante su número. Por ejemplo, en el texto de una determinada sentencia se hace referencia al "artículo 350 inciso 2 del Código General del Proceso", y luego, en otro segmento de la misma sentencia, se hace referencia al "artículo 350 inciso 2". Esta última referencia será descartada por el sistema debido a que carece de fuente explícita.

I) `Predicción: arts. 7 del Pacto Internacional de Derechos Económicos, Sociales y Culturales`

II) `Predicción: art. 9 del Protocolo a la Convención Americana de San Salvador`

III) `Predicción: art. 9 del Pacto Internacional de Derechos Civiles y Políticos`

IV) `Predicción: artículo 6 de la ley de Unión Concubinaria`

Figura 7.20: Ejemplos descartados incorrectamente Artículo

En la figura 7.20 se pueden ver ejemplos de referencias a Artículos descartadas incorrectamente. La gran mayoría se descartó debido a que no se reconoció a la fuente. En los ejemplos listados se pueden ver distintas fuentes diferentes a las tomadas para desarrollar el sistema. Es de suponer, que si se extendiera el repositorio de fuentes conocidas, o se desarrollara otro método de reconocimiento de las mismas, el número de referencias descartadas incorrectamente tendería a 0.

Por último, falta analizar en qué casos el sistema procesó incorrectamente a referencias de tipo Artículo. Como se vio a lo largo de las secciones anteriores, las referencias a Artículos pueden aparecer en muchos formatos, provocando así que el sistema en muchos casos no reconozca todas sus propiedades. Se pudo observar que muchas veces, el procesamiento era incorrecto debido a incoherencias en la redacción de las sentencias. Un ejemplo de esto se ve en la referencia número I) del listado de ejemplos presentado en 7.21, debido a que se utiliza el término "Inc." correspondiente a "Inciso" para referir a un literal. Este caso es procesado incorrectamente debido a que los incisos son numéricos y no alfabéticos. Otro caso se ve con el ejemplo II), en el cual no queda claro a qué se refiere con el segmento de texto "E INCISO 4 DE LA LEY N 16.343" ya que no cuenta con un número de artículo, y el número de artículo presente en la referencia ya está relacionado con la ley 17930. El ejemplo III) muestra cómo el sistema interpretó al número 4 como un nuevo artículo. Si se siguen las reglas de escritura, esta postura no sería incorrecta, debido a que se utiliza el término "arts" el cual indica pluralidad y junto al artículo 350 se asigna el numeral 2. Igualmente, no está claro a qué se corresponde el número 4, ya que casualmente el artículo 350 cuenta con 5 numerales.

I) Predicción: art. 23 Inc. a de CGP

```
Consulta:
MERGE (s:Sentencia {ficha: '0445-000230/2012', numero: '6', año: '2015',
tribunal_juzgado: 'Tribunal de Apelaciones en lo Civil', turno: '6'})
MERGE (x:Articulo {numero: '23', fuente: 'Código General del Proceso'})
MERGE (l:CuerpoNormativo {nombre: 'Código General del Proceso'})
MERGE (x)-[f:Fuente]->(l)
MERGE (s)-[r:Referencia]->(x)
```

II) Predicción: ART. 313 DE LA LEY N 17.930 E INCISO 4 DE LA LEY N 16.343

```
Consulta:
MERGE (s:Sentencia {ficha: '2-41127/2017', numero: '323', año: '2018',
tribunal_juzgado: 'Suprema Corte de Justicia'})
MERGE (x:Articulo {numero: '313', fuente: 'Ley 17930'})
MERGE (l:Ley {numero: '17930'})
MERGE (x)-[f:Fuente]->(l)
MERGE (s)-[r:Referencia {inciso: '4'}] ->(x)
```

III) Predicción: arts. 350.2 y 4 del Código General del Proceso

```
Consulta 1:
MERGE (s:Sentencia {ficha: '435-384/2011', numero: '404', año: '2011',
tribunal_juzgado: 'Tribunal de Apelaciones de Familia', turno: '2'})
MERGE (x:Articulo {numero: '350', fuente: 'Código General del Proceso'})
MERGE (l:CuerpoNormativo {nombre: 'Código General del Proceso'})
MERGE (x)-[f:Fuente]->(l)
MERGE (s)-[r:Referencia {numeral: '2'}]->(x)
Consulta 2:
MERGE (s:Sentencia {ficha: '435-384/2011', numero: '404', año: '2011',
tribunal_juzgado: 'Tribunal de Apelaciones de Familia', turno: '2'})
MERGE (x:Articulo {numero: '4', fuente: 'Código General del Proceso'})
MERGE (l:CuerpoNormativo {nombre: 'Código General del Proceso'})
MERGE (x)-[f:Fuente]->(l)
MERGE (s)-[r:Referencia]->(x)
```

Figura 7.21: Ejemplos incorrectos Artículo

Finalmente, analizando los resultandos obtenidos para las Sentencias, se puede ver que sólo un 5 % de las referencias fueron marcadas como correctamente descartadas. La mayoría de estos casos tratan de referencias a Sentencias en los cuales está presente el número de la sentencia pero no así el año de la misma. En cuanto a las referencias procesadas incorrectamente, estas se correspondieron al 15.90 % de los ejemplos de este tipo. Junto a los Artículos, las referencias a las Sentencias son las que tienen mayor probabilidad de ser procesadas incorrectamente debido a la cantidad de propiedades definidas para las mismas.

I) Predicción: sentencia N 20/2010, del 5/3/10, dictada por el Sr. Juez Letrado de Adolescentes de Primer Turno

```
Consulta:
MERGE (t:Sentencia {numero: '20', año: '2010', tribunal_juzgado:
'Juzgado Letrado', turno: '1'})
WITH t MATCH (x:Sentencia {numero: '20', año: '2010', tribunal_juzgado:
'Juzgado Letrado', turno: '1'}) WHERE size(keys(x)) = 4
MERGE (s:Sentencia {ficha: '437-2/2010', numero: '215', año: '2010',
tribunal_juzgado: 'Tribunal de Apelaciones de Familia', turno: '1'})
MERGE (s)-[r:Referencia]->(x)
```

II) Predicción: Sentencia N 100 de 7 de agosto de 2006, dictada por el Sr. Juez Ldo. de Primera Instancia de la Costa de 1er. T.

```
Consulta:
MERGE (t:Sentencia {numero: '100', año: '2006', tribunal_juzgado: 'Juzgado
Letrado de Primera Instancia', turno: '1'})
WITH t MATCH (x:Sentencia {numero: '100', año: '2006', tribunal_juzgado:
'Juzgado Letrado de Primera Instancia', turno: '1'})
WHERE size(keys(x)) = 4
MERGE (s:Sentencia {ficha: '177-265/2003', numero: '302', año: '2007',
tribunal_juzgado: 'Tribunal de Apelaciones en lo Penal', turno: '1'})
MERGE (s)-[r:Referencia]->(x)
```

III) Predicción: sentencia N 18 de fecha 29 de marzo de 2007, dictada por la Sra. Juez Letrado de Primera Instancia en lo Civil de Décmio Segundo Turno

```
Consulta:
MERGE (t:Sentencia {numero: '18', año: '2007', tribunal_juzgado: 'Juzgado
Letrado de Primera Instancia en lo Civil', turno: '2'})
WITH t MATCH (x:Sentencia {numero: '18', año: '2007', tribunal_juzgado:
'Juzgado Letrado de Primera Instancia en lo Civil', turno: '2'})
WHERE size(keys(x)) = 4
MERGE (s:Sentencia {ficha: '177-265/2003', numero: '302', año: '2007',
tribunal_juzgado: 'Tribunal de Apelaciones en lo Penal', turno: '1'})
MERGE (s)-[r:Referencia]->(x)
```

Figura 7.22: Ejemplos incorrectos Sentencia

El error cometido en el ejemplo I) de 7.22 es no haber reconocido en su totalidad a la sede emisora de la Sentencia 20 del 2010. Si bien esta se corresponde con el Juzgado Letrado de Adolescentes, el sistema no logró especificar el área del juzgado por lo que únicamente definió a la sede como "Juzgado Letrado". Esto se debe a que en el proceso de desarrollo no se tuvo en cuenta esta especialización de forma individual. Otro tipo de error encontrado se puede ver en el ejemplo II) en el cual la localidad de la sede no está completamente especificada. En el texto se utiliza la expresión "de la Costa", lo cual lleva a pensar que se trata de Ciudad de la Costa, la cual sí es soportada por el sistema. Por último, en el ejemplo III)

se da un error debido a la redacción de la referencia, ya que el número de turno presenta una transposición de dos letras. El sistema falla al reconocer dicho número debido a que estos casos no están contemplados.

En la tabla 7.3 se muestran los resultados de la evaluación desde el punto de vista de las medidas Precision, Recall y medida F1. Para poder calcular estas métricas, se tomaron los siguientes conjuntos para cada objeto legal, teniendo en cuenta las distintas clasificaciones definidas al comienzo de la sección 7.6:

- Verdadero Positivo: Referencias marcadas como correctamente procesadas.
- Verdadero Negativo: Referencias marcadas como correctamente descartadas.
- Falso Positivo: Referencias marcadas como incorrectamente procesadas.
- Falso Negativo: Referencias marcadas como incorrectamente descartadas.

Para los tipos Ley y Ficha la recuperación y precisión fueron muy buenas, lo cual se alínea con lo mencionado antes en este capítulo. En el caso de los Artículos, la precisión y la recuperación son casi idénticas debido a que la cantidad de referencias marcadas como incorrectamente procesadas y la cantidad marcada como incorrectamente descartadas difieren únicamente en 1. Por otro lado, para las Sentencias se nota una diferencia mayor. Estos valores se explican por el gran grado de recuperación del sistema, ya que la cantidad de referencias descartadas incorrectamente es pequeña, lo cual indica que el sistema reconoció, en la mayoría de los casos, a las propiedades necesarias para procesar una Sentencia. Sin embargo, se introdujeron bastantes casos de procesamientos incorrectos lo cual provocó la disminución de la precisión. Finalmente, para el objeto legal Decreto, el grado de recuperación resultó bajo en comparación con la precisión debido a la cantidad de ejemplos similares al mencionado anteriormente, el cual no fue soportado por el sistema.

	Precision	Recall	F1-Score
Artículo	94.84	94.77	94.81
Sentencia	83.04	98.45	90.09
Decreto	89.19	70.21	78.57
Ley	100	99.38	99.69
Ficha	100	100	100
Micro-Avg	93.15	96.49	94.79

Tabla 7.3: Resultados evaluación individual

7.6.2. Evaluación del Sistema completo

Más allá de realizar evaluaciones individuales en cada etapa de este trabajo, resulta de interés y se cree necesario el poder tener una noción general del rendimiento del sistema completo. Es decir, realizar una evaluación incluyendo las diferentes etapas del proyecto (Extracción de referencias y Grafo de referencias). Esto se decidió debido a que las evaluaciones individuales no reflejan el error general de todo el proceso. Por ejemplo, evaluar el sistema de carga de información a la base de datos de grafo, no tiene en cuenta el error

cometido por el modelo de extracción de referencias, el cual fue evaluado individualmente en la sección 5.3.2.

Por lo tanto, siguiendo en la línea de las evaluaciones anteriores, para la evaluación completa también fue utilizado el conjunto de evaluación de 200 sentencias. La idea de esta evaluación consistió en ejecutar el sistema completo como se ve en la figura 7.2, es decir, haciendo que cada sentencia incluida en el conjunto de evaluación y almacenada en la BJN, pase por todas las etapas de procesamiento y termine impactada en el grafo de referencias. La metodología de evaluación tomada en este caso, fue comparar únicamente las entidades de una sentencia, con la salida del sistema para cada una de ellas, ignorando el comportamiento de los módulos interiores. Es decir, que si una entidad anotada fue procesada con errores por SpaCy, ya sea porque el modelo extrajo algún token de más o alguno de menos, pero igualmente fue cargada correctamente en el grafo, el caso será tenido en cuenta como correcto. Por otro lado, si a pesar de que el modelo haya recuperado correctamente la referencia anotada, ésta se carga con errores al grafo, será tomado como un caso incorrecto.

Para tener en cuenta el error total cometido por el sistema, en cada iteración fue necesario contar con las referencias anotadas correspondientes a la sentencia procesada en dicho paso. Esto fue necesario para poder reconocer aquellos casos en los cuales el modelo SpaCy no extrajo una entidad y por lo tanto no fue procesada ni cargada al grafo. Luego, la evaluación fue realizada de forma manual teniendo en cuenta al momento de clasificar el procesamiento de una referencia, las mismas opciones que en la evaluación individual del grafo. En la figura 7.23 se presentan diferentes casos presentados a lo largo de la evaluación. En dichos ejemplos se indica el segmento de texto correspondiente a una referencia anotada manualmente, el segmento de texto, cuando corresponda, predicho por el modelo SpaCy, y las consultas generadas por el módulo de carga al grafo.

El ejemplo I) muestra lo que sería un ejemplo correcto de comienzo a fin ya que el modelo SpaCy predijo en su totalidad al texto de la referencia y luego la información de la misma fue cargada correctamente al grafo. Otro ejemplo que se dio durante la evaluación es el presentado en el punto II), donde se puede ver que el modelo SpaCy predijo dos anotaciones de Artículo como una única entidad de este tipo. Esto le generó problemas al módulo de carga al grafo debido a que, como se mencionó en la sección 7.3, al tener cada anotación de Artículos una única fuente, se asumió al extraer las propiedades de una referencia a un objeto legal de este tipo, que la misma contendría únicamente una fuente. Además, la fuente correspondiente al "Pacto Internacional de Derechos Civiles y Políticos" no se encontraba dentro de la lista de fuentes conocidas. Como resultado, se crearon dos nodos de tipo Artículo con fuente Constitución. En este caso, se contó como correcto el procesamiento de la referencia "26 Consitución" (VP), y como incorrecto (FP) el procesamiento de la referencia "10.3 PIDCP". El ejemplo III) se corresponde con un Falso Positivo de SpaCy, ya que el texto "numerales 9 y 10" no fue anotado como una entidad. En este caso el módulo de procesamiento y carga reconoció que no se trataba de una referencia a Artículo y la descartó. Por lo tanto, este caso fue clasificado como descartado correctamente, ya que es información que al final del proceso no fue cargada al grafo.

I) `Anotación: art.1o. inc. 2o. literales A, B, y C de la Ley No. 16.011`

`Predicción: Correcta`

`Consulta:`

```
MERGE (s:Sentencia {ficha: '440-395/2006', numero: '164', año: '2006',
tribunal_juzgado: 'Tribunal de Apelaciones en lo Civil', turno: '2'})
```

```
MERGE (x:Articulo {numero: '1', fuente: 'Ley 16011'})
MERGE (l:Ley {numero: '16011'})
MERGE (x)-[f:Fuente]->(l)
MERGE (s)-[r:Referencia {inciso: '2', literal: 'C,B,A'}]->(x)
```

II) Anotación 1: 10.3 PIDCP

Anotación 2: 26 Constitución

Predicción: 10.3 PIDCP, 26 Constitución,

Consulta 1:

```
MERGE (s:Sentencia {ficha: '89-297/2011', numero: '190', año: '2013',
tribunal_juzgado: 'Tribunal de Apelaciones en lo Penal', turno: '1'})
MERGE (x:Articulo {numero: '10', fuente: 'Constitución'})
MERGE (l:CuerpoNormativo {nombre: 'Constitución'})
MERGE (x)-[f:Fuente]->(l)
MERGE (s)-[r:Referencia {numeral: '3'}]->(x)
```

Consulta 2:

```
MERGE (s:Sentencia {ficha: '89-297/2011', numero: '190', año: '2013',
tribunal_juzgado: 'Tribunal de Apelaciones en lo Penal', turno: '1'})
MERGE (x:Articulo {numero: '26', fuente: 'Constitución'})
MERGE (l:CuerpoNormativo {nombre: 'Constitución'})
MERGE (x)-[f:Fuente]->(l)
MERGE (s)-[r:Referencia {numeral: '3'}]->(x)
```

III) Anotación: N/A

Predicción: numerales 9 y 10

Consulta: Vacía

IV) Anotación: Sent. N 45 de 29/3/11

Predicción: Sent. N 45 de 29/3/11, dictada por la Dra. María Helena Mainard con intervención de la Sra. Fiscal Ltda. Dptal

Consulta:

```
MERGE (t:Sentencia {numero: '45', año: '2011'})
WITH t MATCH (x:Sentencia {numero: '45', año: '2011'})
WHERE size(keys(x)) = 2
MERGE (s:Sentencia {ficha: '457-291/2009', numero: '27', año: '2012',
tribunal_juzgado: 'Tribunal de Apelaciones en lo Penal', turno: '1'})
MERGE (s)-[r:Referencia]->(x)
```

V) Anotación: art. 255 inc. 2 del C.P.P.

Predicción: art. 255 inc. 2 del C.P.P.).- V

Consulta:

```
MERGE (s:Sentencia {ficha: '178-245/2010', numero: '30', año: '2012',
tribunal_juzgado: 'Tribunal de Apelaciones en lo Penal', turno: '4'})
MERGE (x:Articulo {numero: '255', fuente: 'Código del Proceso Penal'})
```

```
MERGE (l:CuerpoNormativo {nombre: 'Código del Proceso Penal'})
MERGE (x)-[f:Fuente]->(l)
MERGE (s)-[r:Referencia {inciso: '2'}]->(x)
```

VI) `Anotación: arts. 8, 537 nal. 5, 688, 841, 842 nal. 5, 1261, 1291 inc. 2, 1296, 1298, 1301, 1308, 1319, 1321, 1324, 1326, 1331 inc. 1o., 1341, 1342, 1405, 1510, 1555, 1560, 1561, 1601 nal. 1o., 1605, 1657, 1660, 1678, 1834, 1897, 1952, 1974, 1991, 2176, 2338, 2372 y concordantes del C.C.`

`Predicción: arts. 8, 537 nal. 5, 688, 841, 842 nal. 5, 1261, 1291 inc. 2, 1296, 1298, 1301, 1308, 1319, 1321`

Figura 7.23: Ejemplos de evaluación completa

Los ejemplos IV) y V) se corresponden a escenarios donde el modelo SpaCy extrajo más texto del anotado. En el primer caso, el modelo extrajo también información acerca de la persona que emitió la Sentencia, mientras que en el segundo caso el modelo extrajo símbolos de más. En la evaluación individual de SpaCy, estos casos fueron tomados como FP, debido a que no se corresponden totalmente con las entidades anotadas. Sin embargo, como se puede ver en las consultas generadas para los ejemplos IV) y V), el módulo de procesamiento y carga al grafo extrajo correctamente las propiedades de las referencias y por lo tanto persistió la información correcta en el grafo. Debido a esto, en el contexto de la evaluación total del sistema, estos casos fueron clasificados como correctos (VP). Finalmente, en el ejemplo VI) se muestra un caso de una referencia a Artículos la cual el modelo SpaCy no extrajo completamente. Este tipo de referencias generalmente se da al final de las Sentencias, y se utiliza como resumen de lo utilizado para la argumentación de la decisión final. Debido a que el modelo no extrajo la fuente de los Artículos, el módulo de procesamiento y carga descartó a la referencia. En este caso, el FP de SpaCy se transformó en un ejemplo descartado incorrectamente (FN) ya que en la anotación se contaba con todas las propiedades necesarias para procesar a la referencia.

	Correcto	Incorrecto	Descartado Correcto	Descartado Incorrecto	**Total**
Artículo	1131	92	196	153	1572
Sentencia	519	145	47	37	748
Decreto	30	5	33	11	79
Ley	128	11	2	34	175
Ficha	229	4	23	9	265
Total	2037	257	301	244	2839
Porcentaje	71.75 %	9.05 %	10.60 %	8.60 %	100 %

Tabla 7.4: Cantidades de referencias en evaluación completa

Viendo las cantidades de referencias clasificadas con cada categoría, se puede apreciar una diferencia con respecto a las cantidades totales de la evaluación individual presentadas en la tabla 7.2. La primera razón de esto es que en la evaluación individual se procesaron la totalidad de las referencias anotadas, incluyendo a aquellas referencias superpuestas, mientras que en la evaluación completa esto no es posible debido

a las restricciones que presentan los sistemas de NER. Por ejemplo, el texto "artículo 2 de la ley 19090" contiene dos anotaciones en el conjunto de evaluación, una para el artículo completo y la otra para la ley. Sin embargo, para procesarla con el modelo SpaCy, la referencia a la ley debió ser quitada del conjunto de entrenamiento. Otro factor influyente se da por los FP de SpaCy, ya que muchos de estos (ejemplo III) presentado anteriormente) terminan siendo descartados por el sistema de procesamiento y carga al grafo.

En la tabla 7.5 se presentan los resultados finales de la evaluación del sistema completo. Para el objeto legal Artículo, se obtuvo un nivel de recuperación de 88.08 %, lo cual muestra una disminución si se toma en cuenta lo obtenido por la evaluación individual del grafo y los resultados obtenidos por el modelo SpaCy. Una de las razones de esto es el aumento de la relación falsos negativos sobre total de referencias, lo cual a su vez es causado por escenarios como los descriptos en los ejemplos II) y VI) comentados anteriormente. En cuanto a las Sentencias, se obtuvo un buen nivel de recuperación. Esto se explica ya que al ser un objeto legal cuyas referencias abarcan segmentos de textos largos, es más probable que el modelo NER de SpaCy cometa errores que terminen en FN. Sin embargo, como se demostró en los ejemplos anteriores, el sistema de procesamiento y carga no se ve afectado en la mayoría de los casos en los cuales el modelo extrae más información de la necesaria, transformando esos FN en VP al final del proceso. Por otro lado, se obtuvo una baja precisión debido a casos de FP similares a los explicados en 7.22. Para los objetos legales restantes se obtuvieron buenos valores de precisión, debido a que como se mencionó en secciones anteriores, las referencias a estos objetos generalmente son concisas y explícitas, lo cual hace más fácil su procesamiento. En cuanto a la recuperación, se puede ver que Decreto y Ley obtuvieron bajos valores, más teniendo en cuenta que la recuperación de SpaCy superó el 92 % en ambos casos. Esto da a entender que el error principal, cuando se habla de recuperación de estos objetos legales, lo cometió el módulo de procesamiento y carga. Finalmente, para el objeto legal Ficha se obtuvieron muy buenos resultados, siguiendo la línea de todas las evaluaciones realizadas anteriormente, ya que es el objeto legal más homogéneo en cuanto a las formas de ser referenciado.

	Precision	Recall	F1-Score
Artículo	92.48	88.08	90.23
Sentencia	78.16	93.35	85.08
Decreto	85.71	73.17	78.95
Ley	92.09	79.01	85.05
Ficha	98.28	96.22	97.24
Micro-Avg	88.80	89.30	89.05

Tabla 7.5: Resultados evaluación completa

Capítulo 8

Conclusiones y trabajo futuro

8.1. Conclusiones

En este proyecto se planteó el objetivo principal de desarrollar un sistema capaz de extraer y clasificar automáticamente referencias de Sentencias de la BJN, y con ellas generar un grafo que facilite la interpretación y manipulación de la información. Para cumplir con esto, se definieron tres etapas destinadas a cumplir con diferentes tareas asociadas tanto a aspectos de PLN como a aspectos de almacenamiento y manipulación de la información.

Como primer resultado de este proyecto se encuentra el corpus anotado sobre 1000 sentencias de la BJN. Debido a la falta de corpus anotados para el idioma Español y destinados a referencias a objetos legales, fue necesario realizar un proceso de anotación manual. De todas formas, el proceso no fue del todo manual ya que para la generación del corpus se utilizaron las Expresiones Regulares como base. Se pudo constatar la gran utilidad de las mismas para detectar patrones habituales en los textos. Sin embargo, si se desea tener un grado de cubrimiento mayor, esta técnica se ve opacada por los métodos de aprendizaje automático los cuales generalizan mejor sobre información aún no vista. Se espera que este aporte resulte de utilidad para continuar trabajando sobre textos en Español relacionados al área legal.

Para la tarea de extraer las referencias a partir de redacciones de sentencias judiciales se experimentó con tres métodos. Se comenzó con NeuroNER debido a que el mismo soportaba la utilización de datos anotados con la herramienta BRAT y, además, estudios realizados demostraron su gran utilidad para las tareas de NER sobre distintas áreas. Con esta herramienta se obtuvieron resultados no satisfactorios cuando se entrenó el modelo sobre todos los objetos legales. Sin embargo, se obtuvieron resultados aceptables al dividir los objetos legales en subconjuntos y realizar diferentes entrenamientos. Luego, se experimentó con un método más simple como lo es CRF, teniendo como objetivo estudiar el peso que conlleva la aparición palabras que identifican al objeto legal. Con este método se obtuvieron resultados similares a los obtenidos con NeuroNER en el entrenamiento por subconjuntos de objetos legales. Finalmente, se realizaron pruebas con la herramienta de NER disponible en el pipeline de SpaCy. Este caso fue con el que se obtuvieron los mejores resultados, y en buenos tiempos de ejecución, generando entonces como salida de todo el proceso

un modelo entrenado capaz de extraer referencias a partir de textos legales. El mejor modelo obtenido alcanzó un 87.63 % de medida micro-F1. El principal aspecto que resultó problemático en el entrenamiento de estos modelos fue la gran variedad de formas de escribir presentes en las Sentencias tratadas, y a su vez el desbalance presente en el corpus utilizado.

Otro recurso generado en este proyecto, fue una lista de 74 Doctrinos conocidos a los cuales se refiere desde las sentencias de la BJN. En el proceso de extracción de los mismos, se pudo confirmar la dificultad de diferenciar de forma automática, y según el contexto, diferentes nombres presentes en el texto. Por lo tanto, se resolvió el problema utilizando diferentes recursos como un sistema basado en reglas para la extracción de fuentes en textos en Español, reglas sobre las categorías gramaticales de las palabras presentes en el contexto de referencias a Doctrinos y restricciones sobre la estructura de las Sentencias.

Un objetivo que no se pudo cumplir fue el de la clasificación semántica de las referencias. Varios fueron los inconvenientes enfrentados en dicho proceso, dentro de los cuales se destaca la falta de conocimiento sobre la jerga utilizada en el ámbito legal, lo cual dificultó el proceso de anotación de predicados. Esto también influyó en que, de todas las Sentencias analizadas en el transcurso del proyecto, no se pudieran identificar muchas categorías semánticas, ya que la mayoría de los predicados se correspondían con una única etiqueta. Es por esto que no se pudo generar un conjunto de categorías a utilizar en una posible clasificación de referencias, y por lo tanto proseguir con esta etapa.

En cuanto al procesamiento de las referencias extraídas y la generación del grafo, se puede afirmar que se obtuvieron resultados satisfactorios. Para esta etapa, al igual que en el caso de la generación del corpus, no se contaban con trabajos previos realizados por lo que se tuvo que implementar desde cero. Para esto se desarrolló un sistema que toma como entrada las entidades extraídas por el modelo NER junto a su etiqueta, procesa el texto de las mismas según el objeto legal que corresponda y finalmente carga la información a una base de datos orientada a grafos. A pesar de no tratarse de un sistema de aprendizaje automático, para su evaluación se siguió la misma línea que en las etapas anteriores, obteniendo un 94.79 % de medida micro-F1.

Como principal recurso generado en este proyecto se encuentra una base de datos orientada a grafos obtenida mediante la ejecución del sistema sobre la totalidad de Sentencias presentes en la BJN. Este grafo contiene un total de 250343 nodos y 1116278 relaciones. Finalmente, fue necesario realizar una evaluación total del sistema, para tener en cuenta los errores cometidos en cada etapa del trabajo. El resultado final obtenido para todo el sistema fue de 89.05 % de medida micro-F1.

8.2. Trabajo futuro

Debido al extenso alcance planteado al principio de este proyecto, sin dudas quedaron muchos aspectos por tratar en cada etapa, y muchos otros por profundizar. Un posible trabajo a futuro sería mejorar el corpus anotado, incluyendo más referencias y un mayor nivel de especificidad de los componentes de las mismas. De esta forma, se podría intentar detectar al mismo tiempo, no sólo el texto correspondiente a una referencia, si no también qué propiedades se encuentran dentro de la misma.

Si bien se obtuvieron buenos resultados en la tapa de extracción automática de referencias, se podría realizar un análisis más profundo de la configuración de hiperparámetros a utilizar, o mismo considerar realizar modificaciones a las arquitecturas de los métodos utilizados. Además, puede ser de utilidad la inclusión de Word Embeddings pre-entrenados sobre los textos de las Sentencias de la BJN. Otro punto importante a tratar, el cual mejoraría el rendimiento del sistema, sería trabajar sobre las anáforas presentes en las redacciones de las sentencias. Esto significó un problema a lo largo de este proyecto, debido a que muchas veces se referencia completamente a un objeto legal, y más adelante se lo referencia nuevamente obviando algunas propiedades, lo cual dificulta el proceso de identificación del mismo si no se cuenta con la información de la referencia anterior.

En cuanto a la extracción de Doctrinos, se podría complementar el proceso realizado con técnicas de aprendizaje automático. El principal problema consiste en diferenciar a los nombres que aparecen en los textos y determinar cuál se corresponde con un Doctrino y cuál no. A su vez, también se podría ampliar la lista obtenida mediante apoyo de expertos del área legal. De esta forma sería posible extender el alcance actual de extracción de referencias a Doctrinos y por lo tanto obtener un grafo más completo. Como primer paso para la extensión del alcance de la extracción de referencias a Doctrinos, se podrían agregar a la lista generada, más registros de formas canónicas de los mismos sumado a varias formas equivalentes para cada nombre, o también expresiones regulares que sirvan para identificar estas variantes. También se podrían utilizar otros componentes del sistema utilizado para esta parte, con el objetivo de relacionar a los Doctrinos con las frases u opiniones emitidas por ellos y presentes en las sentencias.

Un punto a tratar en un futuro es el de la clasificación semántica. Dadas las limitaciones planteadas en la correspondiente sección, se cree necesario el apoyo de entendidos en el tema para la anotación y definición de predicados y categorías semánticas respectivamente. Con estos dos puntos completados, se podrían utilizar métodos supervisados y no supervisados para la categorización de referencias.

Además, para mejorar el rendimiento del módulo de procesamiento y carga de información al grafo, se podrían refinar las reglas hechas con expresiones regulares para obtener las propiedades de cada referencia a partir de los ejemplos con errores obtenidos. Además, algo que mejoraría mucho los resultados para el objeto legal Artículo, sería extender la lista de fuentes conocidas. Otro punto que se podría tratar a futuro sería investigar sobre una herramienta de manipulación del grafo, debido a que Neo4j tiene determinadas limitaciones en cuanto a rendimiento, ya que consume demasiados recursos al intentar visualizar un subgrafo de gran tamaño.

Por último, se cree que lo realizado en este proyecto sirve como base para un futuro desarrollo de una herramienta de visualización, navegación y consulta sobre información de la Base de Jurisprudencia Nacional, la cual permita que la publicación de dicha información sea mejor aprovechada por los usuarios.

Bibliografía

[1] *DB Engines*. https://db-engines.com/en/ranking/graph+dbms. Última vez accedido: 22/01/2020.

[2] Fernandes, Diogo y Jorge Bernardino: *Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB*. páginas 373–380, Enero 2018.

[3] *NeuroNER*. https://neuroner.com. Última vez accedido: 19/02/2020.

[4] Rosá, Aiala: *Identificación de opiniones de diferentes fuentes en textos en español*. mastersphd, PEDECIBA - Universidad de la República, Montevideo, Uruguay - École Doctorale Connaissance, Langage, Modélisation Université Paris Ouest, Nanterre, La Défense, 2011. http://www.fing.edu.uy/inco/grupos/pln/publicaciones/rosa2011.pdf.

[5] Badji, Inés: *Legal entity extraction with NERSystems*. Thesis (Master thesis), E.T.S. de Ingenieros Informáticos (UPM), 2018.

[6] *Base de Jurisprudencia Nacional*. http://bjn.poderjudicial.gub.uy/. Última vez accedido: 14/03/2020.

[7] Sadeghian, A., Sundaram L. Wang D.Z. et al.: *Automatic semantic edge labeling over legal citation graphs*. Artificial Intelligence and Law, 26:127–144, 2018.

[8] Tran, Oanh Thi, Bach Xuan Ngo, Minh Le Nguyen y Akira Shimazu: *Automated Reference Resolution in Legal Texts*. Artif. Intell. Law, página 29–60, 2014.

[9] Adedjouma, Morayo, Mehrdad Sabetzadeh y Lionel C. Briand: *Automated detection and resolution of legal cross references: Approach and a study of Luxembourg's legislation*. 2014 IEEE 22nd International Requirements Engineering Conference (RE), páginas 63–72, 2014.

[10] Leitner, Elena, Georg Rehm y Julián Schneider: *Fine-Grained Named Entity Recognition in Legal Documents*, páginas 272–287. Noviembre 2019, ISBN 978-3-030-33219-8.

[11] Angelidis, Iosif, Ilias Chalkidis y Manolis Koubarakis: *Named Entity Recognition, Linking and Generation for Greek Legislation*. En *JURIX*, 2018.

[12] Konkol, Michal y Miloslav Konopík: *Named Entity Recognition*. PhD Study Report, Západočeská univerzita v Plzni, 2012.

[13] Bell, Sam: *Comparing Graph Databases I*. https://towardsdatascience.com/comparing-graph-databases-5475bdb2e65f. Última vez accedido: 24/11/2019.

[14] Robinson, Ian, Jim Webber y Emil Eifrem: *Graph Databases: New Opportunities for Connected Data*. O'Reilly Media, Inc., 2nd edición, 2015, ISBN 1491930896.

[15] Vicknair, Chad, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen y Dawn Wilkins: *A comparison of a graph database and a relational database: A data provenance perspective*. Volumen 10, página 42, Enero 2010.

[16] Sannier, Nicolas, Morayo Adedjouma, Mehrdad Sabetzadeh y Lionel Briand: *Automated Classification of Legal Cross References Based on Semantic Intent*. En Daneva, Maya y Oscar Pastor (editores): *Requirements Engineering: Foundation for Software Quality*, páginas 119–134. Springer International Publishing, 2016, ISBN 978-3-319-30282-9.

[17] Le, Tuyen Van y H. David Jeong: *NLP-Based Approach to Semantic Classification of Heterogeneous Transportation Asset Data Terminology*. 2017.

[18] Maxwell, Jeremy C., Annie I. Antón y Peter P. Swire: *A legal cross-references taxonomy for identifying conflicting software requirements*. 2011 IEEE 19th International Requirements Engineering Conference, páginas 197–206, 2011.

[19] Van Rossum, Guido y Fred L. Drake: *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009, ISBN 1441412697. `https://www.python.org/`.

[20] Stenetorp, P, Sampo Pyysalo, Goran Topic, Tomoko Ohta, Sophia Ananiadou y Jun'ichi Tsujii: *brat: a Web-based Tool for NLP-Assisted Text Annotation*. páginas 102–107, Abril 2012.

[21] Dernoncourt, Franck, Ji Young Lee y Peter Szolovits: *NeuroNER: an easy-to-use program for named-entity recognition based on neural networks*. Conference on Empirical Methods on Natural Language Processing (EMNLP), 2017.

[22] Martín Abadi, Ashish Agarwal, Paul Barham Eugene Brevdo Zhifeng Chen Craig Citro Greg S Corrado Andy Davis Jeffrey Dean Matthieu Devin et al.: *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, 2015. `http://tensorflow.org/`, Software available from tensorflow.org.

[23] Lafferty, John, Andrew Mccallum y Fernando Pereira: *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data*. Proceedings of the Eighteenth International Conference on Machine Learning, páginas 282–289, Enero 2001.

[24] Tjong Kim Sang, Erik F. y Fien De Meulder: *Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition*. En *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, página 142–147. Association for Computational Linguistics, 2003.

[25] Honnibal, Matthew y Ines Montani: *spaCy: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing*. 2017.

[26] Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard y David McClosky: *The Stanford CoreNLP Natural Language Processing Toolkit*. En *Association for Computational Linguistics (ACL) System Demonstrations*, páginas 55–60, 2014. `http://www.aclweb.org/anthology/P/P14/P14-5010`.

[27] Padró, Lluís y Evgeny Stanilovsky: *FreeLing 3.0: Towards Wider Multilinguality*. En *Proceedings of the Language Resources and Evaluation Conference (LREC 2012)*, Istanbul, Turkey, May 2012. ELRA.

[28] *Amazon Web Services*. `https://aws.amazon.com`. Última vez accedido: 08/09/2019.

[29] *GitHub NeuroNER*. `https://github.com/Franck-Dernoncourt/NeuroNER/issues/3`. Última vez accedido: 09/09/2019.

[30] Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot y E. Duchesnay: *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12:2825–2830, 2011.

[31] *CRFsuite*. `https://sklearn-crfsuite.readthedocs.io`. Última vez accedido: 11/10/2019.

[32] Gupta, Sonal y Christopher Manning: *SPIED: Stanford Pattern based Information Extraction and Diagnostics*. páginas 38–44, Enero 2014.

[33] Gupta, Sonal y Christopher Manning: *Improved Pattern Learning for Bootstrapped Entity Extraction*. páginas 98–108, Enero 2014.

[34] *Damerau-Levenshtein*. `https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-019-2819-0`. Última vez accedido: 09/04/2020.

[35] Sadeghian A, Sundaram L, Wang D Hamilton W Branting K Pfeifer C: *Semantic edge labeling over legal citation graphs*. Proceedings of the workshop on legal text, document, and corpus analytics, páginas 70–75, 2016.

[36] *Ciudades de Uruguay*. `https://viajeaamerica.com/uruguay/ciudades-de-uruguay-geografia-politica`. Última vez accedido: 13/12/2019.

Anexo A. Esquema de anotación BRAT

El siguiente recuadro muestra el esquema de anotación completo para el contexto de las referencias a objetos legales en Sentencias judiciales. Este esquema de anotación se encuentra en el formato requerido por la herramienta BRAT. Se pueden distinguir tres secciones principales, donde la sección "entities" se corresponde con las entidades a anotar, las cuales van desde los objetos legales hasta elementos más específicos incluidos en las referencias donde se destacan los relacionados a Artículos, fechas, números y letras. También se encuentra la sección de atributos, donde se pueden especificar propiedades que pueden tener las entidades. Estas propiedades pueden tomar valores booleanos o enumerados si se configura de esa manera. Para este proyecto únicamente se utilizó un atributo para indicar si una entidad de tipo "número" contenía un sólo número o más de uno.

```
[entities]                          [relations]

Articulo                    <OVERLAP>  Arg1:<ENTITY>, Arg2:<ENTITY>,
Sentencia                   <OVL-TYPE>:<ANY>
Decreto
Ley                         NroArt  Arg1:Articulo, Arg2:Numero
CuerpoNormativo             NroSent Arg1:Sentencia,Arg2:Numero
Doctrino                    NroDecreto  Arg1:Decreto, Arg2:Numero
Ficha                       NroLey  Arg1:Ley, Arg2:Numero
Numero                      NroFicha    Arg1:Ficha, Arg2:Numero
Letra                       SentTurno   Arg1:Sentencia, Arg2:Numero
Fecha                       SentSede    Arg1:Sentencia, Arg2: Sede
Inciso                      Fuente  Arg1:Articulo,
Literal                     Arg2:CuerpoNormativo|Ley|Decreto
Numeral                     ArtNral  Arg1:Numero, Arg2:Numeral
Sede                        ArtInciso   Arg1:Numero, Arg2:Inciso
Turno                       ArtLiteral  Arg1:Numero, Arg2:Literal
                            NroTurno    Arg1:Turno, Arg2:Numero
[attributes]                NroInciso   Arg1:Inciso, Arg2:Numero
                            NroNumeral  Arg1:Numeral, Arg2:Numero
Varios  Arg:Numero          LetraLiteral    Arg1:Literal, Arg2:Letra
                            FechaDe Arg1:<ENTITY>, Arg2:Fecha
```

La última sección es la de relaciones, donde se configuran las posibles relaciones que se pueden dar entre dos entidades. Dentro de estas se destacan las relaciones entre entidades correspondientes a objetos legales y entidades de tipo "número", y la relación "fuente" que es utilizada para relacionar a un Artículo con su fuente de tipo Decreto, Ley o Cuerpo Normativo. Por último, cabe destacar que al comienzo de la sección "relations" se encuentran dos registros utilizados para que la herramienta BRAT permita al usuario realizar anotaciones que se superpongan.

Anexo B. Expresiones Regulares

A continuación se detallan las expresiones regulares creadas y utilizadas para la creación del subcorpus. El primer recuadro contiene las variables definidas para que el proceso de creación fuese más claro e interpretable.

```
# FUENTES
entities = Cuerpos Normativos conocidos

# EXTRAS
extras = palabras clave que ocurren en el contexto de una referencia a
Articulos

# VARIABLES
art = ‘art[í|i]culo|art’
plural = ‘s?’
abrev = ‘\.?’
nums = ‘[,\.\dºº]’
numso = ‘[\s,\.\dººo]’
num_symbol = ‘[ººnro\.]+’
coord = ‘y?’
pre = ‘del|de’
var = extras+‘|m|n|o|y’
numbers = nums+‘+\s*(‘+var+’)?’
det_fem = ‘(?:la)?’
date = ‘(?:/(\d{2,4})|(?:.{0,20}?\d{1,2}.{0,20}?de\s*(\d{2,4}))|.{0,20}?
\d{1,2}(?:/|.)(?:\d{1,2}|.{1,2})(?:/|.)(\d{2,4}))’
connection = ‘([,;y]+)’
range_connector = ‘[a-]’
letter = ‘[a-z]’
words = ‘[a-z\s]’
```

La variable “entities” se corresponde con los Cuerpos Normativos observados en el proceso de creación. Como se mencionó en la sección correspondiente, se utilizaron 100 sentencias para la creación de las expresiones regulares. En dicho proceso, se fue analizando el comportamiento de las mismas sobre cada sentencia, y a su vez se fueron recopilando los Cuerpos Normativos. La variable “extras” contiene palabras relacionadas a Artículos como numeral, inciso, literal y sus derivadas.

```
# REGLAS PARA ARTICULOS
rules = {

    'EANS': '(('+extras+')(?:[])]?)'+numso+'+)+(?:'+pre+')?\s*(?:'+art+')
    ('+plural+')'+abrev+'\s*('+nums+'+)\s*(?:'+pre+')?\s*'+det_fem+'\s*
    ('+source+')',
    'EAN': '(('+extras+')(?:[])]?)\s*'+nums+'*)+\s*(?:'+pre+')?\s*
    (?:'+art+')('+plural+')'+abrev+'\s*('+nums+'+)',
    'NS': '(?:'+connection+')((?:\s*'+numbers+')+)\s*(?:'+pre+')?\s*'+
    det_fem+'\s*('+source+')',
    'AM': '(?:'+art+')('+plural+')'+abrev+'\s(?:'+letter+')\s*
    ((?:'+numbers+')+)\s*(?:'+pre+')?\s*'+det_fem+'\s*('+source+')?',
    'ANS': '(?:'+art+')('+plural+')'+abrev+'((?:\s*'+nums+'*'+
    range_connector+'?('+var+'|lro|to)?'+')+)\s*(?:'+pre+')?\s*'+det_fem+
    '\s*('+source+')',
    'ANWS': '(?:'+art+')('+plural+')'+abrev+'\s*((?:'+nums+'+'+
    range_connector+'?('+var+')?'+')+)\s*(?:'+words+'{0,10})(?:'+pre+')?
    \s*'+det_fem+'\s*('+source+')',
    'SAN': '('+source+')\s*,?\s*\(?\s*(?:'+art+')('+plural+')'+abrev+
    '((?:\s*'+nums+'*'+range_connector+'?('+var+')?'+')+)',
    'ANPS': '(?:'+art+')('+plural+')'+abrev+'(((?:\s*'+nums+'*y?'+
    range_connector+'?)+(\((\s*('+var+'|lro|to|primero|segundo)?'+nums+'*)
    +\)))+)\s*(?:'+pre+')?\s*'+det_fem+'\s*('+source+')',
    'AN': '(?:'+art+')('+plural+')'+abrev+'((?:\s*'+nums+'+\s*'+
    range_connector+'?('+var+')?'+')+)'
}

# ARTICULOS
article = '|'.join('(?P<{rule_name}>{regex})'
             .format(rule_name=name,regex=expr)
             for name, expr in rules.items())
```

En el recuadro de Artículos se ven las reglas mencionadas en la sección 4.1.1, las cuales utilizan las variables definidas en el primer recuadro y están ordenadas según su nivel de especificidad. Luego, se ve el método utilizado para crear una única expresión regular encargada de extraer los artículos aplicando desde la regla más específica a la más general.

En el siguiente recuadro se pueden observar las expresiones regulares para Ley y Decreto. Además de la expresión correspondiente a los números de una Ley, se crearon dos reglas para cada tipo de objeto legal. Una de ellas para abarcar los casos singulares, es decir cuando se espera un única número, mientras que la otra se formuló con la intención de abarcar los casos plurales.

```
# LEYES
law_num =  ‘(?:\d\s*(?:,|\.)?\s*)’

law = ‘ley\s*(?:’+num_symbol+‘)?\s*(’+law_num+‘+)\s*(?:(?:’+pre+‘)\s*(‘+date+’))?’

laws = ‘leyes(?:\s*(?:,|;|y)?\s*(?:Nros?\.?|Nos?\.?|N\s*º?)?\s*(?:\d{3,5}))+’

# DECRETOS
decree = ‘(decreto|decreto-ley|d-ley|dec-ley|dec\.|dl)\s*([^\d\.]+)?\s*(?:Nro\.?|No\.?
|N\s*º?)?\s*(?:\d{1,2}\.\d{3}|\d{2,5})(?:(?:/|-)(\d{2,7})|‘+ date + ’)?’

decrees = ‘(decretos|decretos-leyes|dec-leyes)(?:\s*(?:,|;|y)?\s*(?:Nros?\.?|Nos?\.?|
N\s*º?)?\s*(?:\d{2,4}|\d{1}\.\d{3})(?:/\d{2,5})?)+’
```

Por último, se presentan las expresiones utilizadas para Sentencia y Ficha. En el caso de Sentencia se desarrollaron reglas para identificar las palabras clave, ya sean en singular o plural, la sede de la sentencia y reglas para abarcar tanto referencias en singular como en plural. Por otro lado, para el caso de las Fichas, se utilizó únicamente una expresión regular.

```
# SENTENCIAS
sent_name = ‘(sentencia|sent\.|resolución|providencia)’

sents_name = ‘(sentencias|sents\.|resoluciones|providencias)’

date_num = ‘\d{1,4}(?:/\d{2,4})?’

court = ‘(Jueza.{0,100}?\s+(?:Turno|T\.)|Juez.{0,100}?\s+(?:Turno|T\.)|Juzgado.{0,100}
?\s+(?:Turno|T\.)|Tribunal.{0,100}?\s+(?:Turno|T\.)|Dr\..{0,100}?\s+(?:Turno|T\.))’

sentence = sent_name + ‘\s*(?:\w{0,15}?\s){0,4}?(?:Nro\.?|No\.?|N[º°]?)?\s*(\d{1,4})’
+ date + ‘?(?:.{0,100}’ + court + ‘)?’

sentences = sents_name + ‘\s*.{0,40}?,?(?:\s*(?:Nros?\.?|Nos?\.?|N[º°]?)?\s*‘ +
date_num + ’\s*(?:,|;|y)?)+’

# FICHAS
case_file = ‘(?<=[^\w])(ficha|Fª|Fa?\.?|I\.?U\.?E\.?|expediente|exp?\.?)\s*:?-?\s*
(?:Nro\.?|No\.?|N\s*(?:º|°)?)?\s*(\d+\s*(?:\.|-|{|/)?\s*\d*\.?\d*)\s*
(?:/|-)\s*(\d\.\d{3}|\d{2,4})’
```

Anexo C. Extracción de Referencias

Ejemplos de extracción con SpaCy

A continuación se pueden ver ejemplos de comparaciones entre anotaciones y extracciones mediante el modelo entrenado de SpaCy. Una primera observación es que al parecer cuanto más extenso es el texto de la referencia anotada, más probable es que el modelo cometa algún error. Entre los errores cometidos en dichos casos se destacan los que se dan al no extraer alguna parte final del texto de la referencia. Esto se puede ver en los ejemplos III a V, inclusive. Otra razón por la cual se dan estos errores es que, las fuentes de los Artículos son demasiado extensas y poco frecuentes en el corpus. Luego, hay ejemplos similares al IX y X, donde SpaCy no extrae una anotación compuesta, si no que la misma es dividida y extraída en partes. También existen situaciones donde el texto predicho excede al anotado, como en los ejemplos VIII, XIII y XVII y situaciones donde la anotación no es predicha ni parcialmente como el caso XV.

I) `Anotación: art. 203.4 in fine y 204.2 CGP`

   `Predicción: Correcta`

II) `Anotación: arts. 56 [red. L. 19090] y 261 [red. L. 16699] CGP`

   `Predicción: Correcta`

III) `Anotación: art. 14.1 del Pacto Internacional de Derechos Civiles y políticos de 16 de diciembrede 1966`

   `Predicción: art. 14.1 del Pacto Internacional de Derechos Civiles`

IV) `Anotación: art. 8 de la Convención Americana sobre Derechos Humanos`

   `Predicción: art. 8 de la Convención Americana`

V) `Anotación: artículos 34 a 38 inclusive del Tratado de Derecho Civil de 1889`

   `Predicción: artículos 34 a 38`

VI) `Anotación: art. 341 de la ley Nº 18.172 de fecha 31 de agosto de 2007`

   `Predicción: art. 341 de la ley Nº 18.172`

VII) `Anotación: Convención Sobre los Derechos del Niño (arts. 8 y 9)`

   `Predicción: arts. 8 y 9`

VIII) `Anotación: art. 200.1 num.1 del C.G.P.`

   `Predicción: art. 200.1 num.1 del C.G.P.- C O N`

IX) `Anotación: artículos 2 lit. C) y art. 33 del Título 7 del Texto Ordenado 1996`

   `Predicción: artículos 2 lit. C`

X) `Anotación: arts. 5 in fine, 48 in fine, 69, 85 nal. 4 y 87`

   `Predicción: arts. 5`

XI) `Anotación: Sentencia de la Sala Nro. 199 del 4/11/1997`

   `Predicción: Sentencia de la Sala Nro. 199`

XII) Anotación: SENTENCIA Nº 304 Montevideo, nueve de diciembre de dos mil once. TRIBUNAL DE APELACIONES EN LO CIVIL DE CUARTO TURNO

Predicción: Correcta

XIII) Anotación: providencia Nro. 4/2010

Predicción: providencia Nro. 4/2010 dictada por la Sra. Jueza de Feria

XIV) Anotación: N/A

Predicción: Sent. del Tribunal Supremo del 31.1.92

XV) Anotación: TAT 2 Sentencia 174/2008

Predicción: N/A

XVI) Anotación: Dec-Ley Nº 14.188 de 5 de abril de 1974

Predicción: Correcta

XVII) Anotación: Ley 19090

Predicción: Ley 19090-

XVIII) Anotación: Ficha 469 1116 2009

Predicción: Correcta

XIX) Anotación: IUE: 0002 016968/2013

Predicción: Correcta

Anexo D. Grafo de Referencias

Ejemplos del módulo de procesamiento y carga al Grafo

En esta sección se presentan ejemplos complementarios de la evaluación individual realizada sobre el módulo de procesamiento y carga de información al grafo de referencias. A destacar se pueden identificar los casos de Artículos para los cuales no se reconoció la fuente y por lo tanto fueron descartados incorrectamente, como los ejemplos IV y V. Luego, para este objeto legal, se pueden ver ejemplos incorrectos que resultan interesantes, como lo son los VII, IX, X y XI. El primero muestra un error del sistema al tomar a los literales como pertenecientes a un único Artículo, mientras que el error del ejemplo IX surge debido a la nomenclatura utilizada para indicar el párrafo del Artículo. El sistema falla en este caso debido a que no se contemplan referencias a párrafos, si no que se contemplan referencias a incisos, numerales y literales. En cuanto al ejemplo X, se presenta el token "350.2.4" lo cual genera confusión acerca de su significado, ya que usualmente se utiliza un único número luego del punto indicando el numeral del Artículo. Por último, el ejemplo XI presenta el token "60-1", lo cual genera error ya que el sistema se desarrolló tokenizando y aplicando reglas sobre cada token, y el token presentado no contiene un formato usual.

I) `Anotación: inciso 2 del artículo 255 del Código del Proceso Penal`

```
Consulta:
MERGE (s:Sentencia {ficha: ‘369-203/2010’, numero: ‘81’, año: ‘2011’,
tribunal_juzgado: ‘Tribunal de Apelaciones en lo Penal’, turno: ‘4’})
MERGE (x:Articulo {numero: ‘255’, fuente: ‘Código del Proceso Penal’})
MERGE (l:CuerpoNormativo {nombre: ‘Código del Proceso Penal’})
MERGE (x)-[f:Fuente]->(l)
MERGE (s)-[r:Referencia {inciso: ’2’}]->(x)
```

`Clasificación: Correcto`

II) `Anotación: art. 1 literales ‘‘b’’ y ‘‘c’’ de la Ley N 15.881`

```
Consulta:
MERGE (s:Sentencia {ficha: ‘2-21267/2010’, numero: ‘250’, año: ‘2011’,
tribunal_juzgado: ‘Tribunal de Apelaciones en lo Civil’, turno: ‘5’})
MERGE (x:Articulo {numero: ‘1’, fuente: ‘Ley 15881’})
MERGE (l:Ley {numero: ‘15881’})
MERGE (x)-[f:Fuente]->(l)
MERGE (s)-[r:Referencia {literal: ‘c,b’}]->(x)
```

`Clasificación: Correcto`

III) `Anotación: C.G.P. art.119.1`

```
Consulta:
MERGE (s:Sentencia {ficha: ‘191-25/2010’, numero: ‘237’, año: ‘2011’,
tribunal_juzgado: ‘Tribunal de Apelaciones en lo Civil’, turno: ‘1’})
MERGE (x:Articulo {numero: ‘119’, fuente: ‘Código General del Proceso’})
```

```
MERGE (l:CuerpoNormativo nombre: 'Código General del Proceso')
MERGE (x)-[f:Fuente]->(l)
MERGE (s)-[r:Referencia {numeral: '1'}]->(x)
```

Clasificación: Correcto

IV) Anotación: art. 1 del C.I.T. 95

Clasificación: Descartado Incorrecto

V) Anotación: arts. 107 y ss. del Presupuesto Quinquenal 2006/2010

Clasificación: Descartado Incorrecto

VI) Anotación: leyes 15.361 (artículo 5)

Consulta:

```
MERGE (s:Sentencia {año: '2010', ficha: '432 347 2006', numero: '281',
tribunal_juzgado: 'Tribunal de Apelaciones en lo Civil', turno: '2'})
MERGE (x:Articulo {numero: '5', fuente: 'Ley 15361'})
MERGE (l:Ley {numero: '15361'})
MERGE (x)-[f:Fuente]->(l)
MERGE (s)-[r:Referencia]->(x)
```

Clasificación: Correcto

VII) Anotación: arts. 3 lit. ''B'' y ''D'', 7 lit. ''G'', 20, 21, 27 lit. ''B'' y ''C'' y 39 lit. ''J'' y concordantes de la Ley No. 16.696

Consulta:

```
MERGE (s:Sentencia {año: '2010', ficha: '22-1406/2004', numero: '846',
tribunal_juzgado: 'Suprema Corte de Justicia'})
MERGE (x:Articulo {numero: '3', fuente: 'Ley 16696'})
MERGE (l:Ley {numero: '16696'})
MERGE (x)-[f:Fuente]->(l)
MERGE (s)-[r:Referencia {literal: 'D,G,C,J,B'}]->(x)
```

Clasificación: Incorrecto

VIII) Anotación: Dec. reglamentario No. 244/00 (art. 18)

Consulta:

```
MERGE (s:Sentencia {año: '2010', ficha: '22-1406/2004', numero: '846',
tribunal_juzgado: 'Suprema Corte de Justicia'})
MERGE (x:Articulo {numero: '18', fuente: 'Decreto 244 de 2000'})
MERGE (l:Decreto {numero: '244', año: '2000'})
MERGE (x)-[f:Fuente]->(l)
MERGE (s)-[r:Referencia]->(x)
```

Clasificación: Correcto

IX) Anotación: a. 197, 4to. pfo. C.G.P.

Consulta 1:

```
MERGE (s:Sentencia {año: '2011', ficha: '41-21/2010', numero: '390',
```

```
tribunal_juzgado: 'Tribunal de Apelaciones en lo Civil', turno: '2'})
MERGE (x:Articulo {numero: '197', fuente: 'Código General del Proceso'})
MERGE (l:CuerpoNormativo {nombre: 'Código General del Proceso'})
MERGE (x)-[f:Fuente]->(l)
MERGE (s)-[r:Referencia]->(x)
```

```
Consulta 2:
MERGE (s:Sentencia {año: '2011', ficha: '41-21/2010', numero: '390',
tribunal_juzgado: 'Tribunal de Apelaciones en lo Civil', turno: '2'})
MERGE (x:Articulo {numero: '4', fuente: 'Código General del Proceso'})
MERGE (l:CuerpoNormativo {nombre: 'Código General del Proceso'})
MERGE (x)-[f:Fuente]->(l)
MERGE (s)-[r:Referencia]->(x)
```

```
Clasificación: Incorrecto
```

X) Anotación: art. 56, 261 y 350.2.4 del Código General del Proceso

```
Consulta 1:
MERGE (s:Sentencia {año: '2013', ficha: '341-326/2011', numero: '11',
tribunal_juzgado: 'Tribunal de Apelaciones de Familia', turno: '2'})
MERGE (x:Articulo {numero: '56', fuente: 'Código General del Proceso'})
MERGE (l:CuerpoNormativo {nombre: 'Código General del Proceso'})
MERGE (x)-[f:Fuente]->(l)
MERGE (s)-[r:Referencia]->(x)
```

```
Consulta 2:
MERGE (s:Sentencia {año: '2013', ficha: '341-326/2011', numero: '11',
tribunal_juzgado: 'Tribunal de Apelaciones de Familia', turno: '2'})
MERGE (x:Articulo {numero: '261', fuente: 'Código General del Proceso'})
MERGE (l:CuerpoNormativo {nombre: 'Código General del Proceso'})
MERGE (x)-[f:Fuente]->(l)
MERGE (s)-[r:Referencia]->(x)
```

```
Clasificación: Incorrecto
```

XI) Anotación: arts. 18, 60-1, 312 num. 4° CP

```
Consulta 1:
MERGE (s:Sentencia {año: '2017', ficha: '375-30/2014', numero: '172',
tribunal_juzgado: 'Tribunal de Apelaciones en lo Penal', turno: '1'})
MERGE (x:Articulo {numero: '18', fuente: 'Código Penal'})
MERGE (l:CuerpoNormativo {nombre: 'Código Penal'})
MERGE (x)-[f:Fuente]->(l)
MERGE (s)-[r:Referencia]->(x)
```

```
Consulta 2:
MERGE (s:Sentencia {año: '2017', ficha: '375-30/2014', numero: '172',
tribunal_juzgado: 'Tribunal de Apelaciones en lo Penal', turno: '1'})
MERGE (x:Articulo {numero: '312', fuente: 'Código Penal'})
```

```
MERGE (l:CuerpoNormativo {nombre: 'Código Penal'})
MERGE (x)-[f:Fuente]->(l)
MERGE (s)-[r:Referencia {numeral: '4'}]->(x)
```

Clasificación: Incorrecto

XII) Anotación: art. 61 inc. III de la Ley 15.750

Consulta:

```
MERGE (s:Sentencia {año: '2014', ficha: '0476-000002/2014', numero: '8',
tribunal_juzgado: 'Tribunal de Apelaciones en lo Civil', turno: '7'})
MERGE (x:Articulo {numero: '61', fuente: 'Ley 15750'})
MERGE (l:Ley {numero: '15750'})
MERGE (x)-[f:Fuente]->(l)
MERGE (s)-[r:Referencia]->(x)
```

Clasificación: Incorrecto

XIII) Anotación: art. 56 de la Ley 18083

Consulta:

```
MERGE (s:Sentencia {año: '2011', ficha: '445/103/2011', numero: '229',
tribunal_juzgado: 'Tribunal de Apelaciones en lo Civil', turno: '2'})
MERGE (x:Articulo {numero: '56', fuente: 'Ley 18083'})
MERGE (l:Ley {numero: '18083'})
MERGE (x)-[f:Fuente]->(l)
MERGE (s)-[r:Referencia]->(x)
```

Clasificación: Correcto

En cuanto a los ejemplos de Sentencias presentados a continuación, se detaca como incorrecto el ejemplo XV. El mismo resulta incorrecto debido a que la propiedad "tribunal_juzgado" correspondiente a la sede emisora de la Sentencia fue mal determinada. En este caso el problema surge a que la sede presenta una especificación del área, "Especializado en Crimen Organizado", lo cual no fue tenido en cuenta a la hora del desarrollo del sistema. Luego, se pueden ver varios ejemplos procesados correctamente y uno descartado de manera correcta. Este último, el ejemplo XIV, fue descartado correctamente debido a la falta del año en la referencia.

XIV) Anotación: No.279 Tribunal de Apelaciones en lo Civil de Segundo Turno

Clasificación: Descartado Correcto

XV) Anotación: sentencia definitiva N 12 del 17 de agosto de 2012, dictada por el Sra. Juez Letrado de 1a. Instancia en lo Penal Especializado en Crimen Organizado de 2 Turno

Consulta:

```
MERGE(t:Sentencia {numero: '12', año: '2012', tribunal_juzgado:
'Juzgado Letrado de Primera Instancia en lo Penal', turno: '2'})
WITH t MATCH(x:Sentencia {numero: '12', año: '2012', tribunal_juzgado:
'Juzgado Letrado de Primera Instancia en lo Penal', turno: '2'})
```

```
WHERE size(keys(x)) = 4
MERGE (s:Sentencia {ficha: '475-87/2009', numero: '295', año: '2013',
tribunal_juzgado: 'Tribunal de Apelaciones en lo Penal', turno: '4'})
MERGE (s)-[r:Referencia]->(x)
```

Clasificación: Incorrecto

XVI) Anotación: T.A.C. 8o. Sent. No. 162/92

```
Consulta:
MERGE(t:Sentencia {numero: '162', año: '1992', tribunal_juzgado:
'Tribunal de Apelaciones en lo Civil', turno: '8'})
WITH t MATCH(x:Sentencia {numero: '162', año: '1992', tribunal_juzgado:
'Tribunal de Apelaciones en lo Civil', turno: '8'})
WHERE size(keys(x)) = 4
MERGE (s:Sentencia {ficha: '22-1406/2004', numero: '846', año: '2010',
tribunal_juzgado: 'Suprema Corte de Justicia'})
MERGE (s)-[r:Referencia]->(x)
```

Clasificación: Correcto

XVII) Anotación: SCJ en S 20/1992

```
Consulta:
MERGE(t:Sentencia {numero: '20', año: '1992', tribunal_juzgado:
'Suprema Corte de Justicia'})
WITH t MATCH(x:Sentencia {numero: '20', año: '1992', tribunal_juzgado:
'Suprema Corte de Justicia'})
WHERE size(keys(x)) = 3
MERGE (s:Sentencia {año: '2010', ficha: '432 347 2006', numero: '281',
tribunal_juzgado: 'Tribunal de Apelaciones en lo Civil', turno: '2'})
MERGE (s)-[r:Referencia]->(x)
```

Clasificación: Correcto

XVIII) Anotación: Sentencia Interlocutoria No. 458 de 3.VIII.2011 dictada por el Tribunal de Apelaciones en lo Civil de Primer Turno

```
Consulta:
MERGE(t:Sentencia {numero: '458', año: '2011', tribunal_juzgado:
'Tribunal de Apelaciones en lo Civil', turno: '1'})
WITH t MATCH (x:Sentencia{numero: '458', año: '2011', tribunal_juzgado:
'Tribunal de Apelaciones en lo Civil', turno: '1'})
WHERE size(keys(x)) = 4
MERGE (s:Sentencia {ficha: '3-5/2011', numero: '1710', año: '2011',
tribunal_juzgado: 'Suprema Corte de Justicia'})
MERGE (s)-[r:Referencia]->(x)
```

Clasificación: Correcto

Los últimos ejemplos presentados en este Anexo se corresponden a ejemplos de los objetos legales Decreto, Ley y Ficha procesados correctamente. Dentro de estos se destacan los relacionados a Ficha, los cuales, junto a las palabras completas "expediente" y "ficha", son los que representan a las formas más comunes de referencias a un objeto de este tipo.

XIX) Anotación: Decreto No. 285 de 1o..IX.2011

```
Consulta:
MERGE(s:Sentencia {ficha: '96-182/2009', numero: '2151', año: '2011',
tribunal_juzgado: 'Suprema Corte de Justicia'})
MERGE(x:Decreto {numero: '285', año: '2011'})
MERGE (s)-[r:Referencia]->(x)
```

Clasificación: Correcto

XX) Anotación: LEY No. 18.572

```
Consulta:
MERGE(s:Sentencia {ficha: '467-48/2010', numero: '469', año: '2010',
tribunal_juzgado: 'Suprema Corte de Justicia'})
MERGE(x:Ley {numero: '18572'})
MERGE (s)-[r:Referencia]->(x)
```

Clasificación: Correcto

XXI) Anotación: I.U.E. 191-25/2010

```
Consulta:
MERGE(s:Sentencia {ficha: '191-25/2010', numero: '237', año: '2011',
tribunal_juzgado: 'Tribunal de Apelaciones en lo Civil', turno: '1'})
MERGE (x:Ficha {numero: '191-25', año: '2010'})
MERGE(s)-[r:Referencia]->(x)
```

Clasificación: Correcto

XXII) Anotación: Ex. N 439-15/2011

```
Consulta:
MERGE(s:Sentencia {ficha: '439-15/2011', numero: '230', año: '2011',
tribunal_juzgado: 'Tribunal de Apelaciones de Familia', turno: '1'})
MERGE (x:Ficha {numero: '439-15', año: '2011'})
MERGE (s)-[r:Referencia]->(x)
```

Clasificación: Correcto

XXIII) Anotación: F. N329/158/09

```
Consulta:
MERGE(s:Sentencia {ficha: '329-158/2009', numero: '300', año: '2011',
tribunal_juzgado: 'Tribunal de Apelaciones del Trabajo', turno: '3'})
MERGE (x:Ficha {numero: '329-158', año: '2009'})
MERGE (s)-[r:Referencia]->(x)
```

Clasificación: Correcto

Anexo E. Lista de Cuerpos Normativos

En este anexo se adjunta la lista de Cuerpos Normativos utilizada como apoyo para el reconocimiento de fuentes en referencias a Artículos.

```
Constitución de la República Oriental del Uruguay | Constitución
Constitución de la República | Constitución
Constitución Nacional | Constitución
Constitución | Constitución
Const. de la República | Constitución
Const. Nacional | Constitución
Const. | Constitución
Reglamento Nacional de Circulación Vial | Reglamento Nacional de Circulación Vial
Reglamento de Circulación Vial | Reglamento Nacional de Circulación Vial
R.N.C.V. | Reglamento Nacional de Circulación Vial
RNCV | Reglamento Nacional de Circulación Vial
Código de la Niñez y Adolescencia | Código de la Niñez y Adolescencia
Código de la Niñez | Código de la Niñez y Adolescencia
Código de la Adolescencia | Código de la Niñez y Adolescencia
C.N.A. | Código de la Niñez y Adolescencia
CNA | Código de la Niñez y Adolescencia
C. del Niño | Código de la Niñez y Adolescencia
CN | Código de la Niñez y Adolescencia
Convenio Internacional de Trabajo | Convenio Internacional de Trabajo
CIT | Convenio Internacional de Trabajo
C.I.T | Convenio Internacional de Trabajo
Convenio OIT | Convenio Internacional de Trabajo
Código de Procedimiento Civil | Código de Procedimiento Civil
CPC | Código de Procedimiento Civil
C.P.C | Código de Procedimiento Civil
Ley de Sociedades Comerciales | Ley de Sociedades Comerciales
Ley Orgánica Policial | Ley Orgánica Policial
Ley Policial | Ley Orgánica Policial
Código General del Proceso | Código General del Proceso
C.G.P. | Código General del Proceso
```

C.G.P | Código General del Proceso

CGP | Código General del Proceso

Código del Proceso Penal | Código del Proceso Penal

C.P.P. | Código del Proceso Penal

CP.P. | Código del Proceso Penal

CPP | Código del Proceso Penal

Código de Minierí a | Código de Minierí a

C.M. | Código de Minierí a

Código Aeronáutico | Código Aeronáutico

C.A. | Código Aeronáutico

Código de Comercio | Código de Comercio

C. de Comercio | Código de Comercio

C. Comercio | Código de Comercio

C. de Com. | Código de Comercio

C. Com. | Código de Comercio

Código Tributario | Código Tributario

C.T. | Código Tribuatrio

Código Rural | Código Rural

C.R. | Código Rural

Código Penal | Código Penal

C. Penal | Código Penal

C.P. | Código Penal

CP | Código Penal

Código Civil | Código Civil

Cód. Civil. | Código Civil

Cód. Civ. | Código Civil

C. Civil | Código Civil

C.C | Código Civil

CC | Código Civil

Carta Magna | Carta

Carta | Carta

Ley Orgánica de la Judicatura y de Organización de los Tribunales | LOT

L.O.T. | LOT

LOT | LOT