UNIVERSIDAD DE LA REPÚBLICA

Extracción de eventos en prensa escrita Uruguaya del siglo XIX

por

Pablo Anzorena

Manuel Laguarda

Bruno Olivera

Tutora:

Regina Motz

Informe de Proyecto de Grado presentado al Tribunal Evaluador
como requisito de graduación de la carrera
Ingeniería en Computación

en la Facultad de Ingeniería

1. Resumen

En este proyecto, se plantea el diseño y la implementación de un sistema de extracción de eventos en prensa uruguaya del siglo XIX digitalizados en formato de imagen, generando clusters de eventos agrupados según su similitud semántica.

La solución propuesta se divide en 4 módulos: módulo de preprocesamiento compuesto por el OCR y un corrector de texto, módulo de extracción de eventos implementado en Python y utilizando Freeling[1], módulo de clustering de eventos implementado en Python utilizando Word Embeddings y por último el módulo de etiquetado de los clusters también utilizando Python.
Debido a la cantidad de ruido en los datos que hay en los diarios antiguos, la evaluación de la solución se hizo sobre datos de prensa digital de la actualidad. Se evaluaron diferentes medidas a lo largo del proceso. Para la extracción de eventos se logró conseguir una Precisión y Recall de un 56% y 70% respectivamente. En el caso del módulo de clustering se evaluaron las medidas de Silhouette Coefficient, la Pureza y la Entropía, dando 0.01, 0.57 y 1.41 respectivamente. Finalmente se etiquetaron los clusters utilizando como etiqueta las secciones de los diarios de la actualidad, realizándose una evaluación del etiquetado.

[1] http://nlp.lsi.upc.edu/freeling/demo/demo.php

Índice general

2. Introducción

2.1. Planteo del problema

El presente proyecto forma parte de un proyecto mayor titulado “Compartiendo la historia escondida del cambio climático en Latinoamérica a través de las TIC”. El mismo está respaldado por el Instituto Panamericano de Geografía e Historia (IPGH) y del cual participan grupos de las siguientes entidades: Pontificia Universidad Javeriana (Colombia), Universidad de Cuenca (Ecuador), Universidad del Azuay (Ecuador), Universidad de las Américas Puebla (México) y Universidad de la República (Uruguay).

Considerando el reconocimiento del problema del cambio climático en Latinoamérica, en el contexto de este proyecto se pretende mirar al pasado para entender mejor el presente. Así, realizaremos una retrospectiva de la cuestión con el objetivo de analizar los antecedentes históricos del cambio climático en el territorio Latinoamericano.

Para esta mirada al pasado a los eventos climáticos ocurridos en Uruguay, se va a realizar un análisis de diarios del siglo XIX mediante técnicas de Procesamiento de Lenguaje Natural (PLN).

La Biblioteca Nacional del Uruguay cuenta con diarios de esta época digitalizados en formato de imagen. La calidad de los mismos varía en las diferentes editoriales y ejemplares, por lo que la correcta conversión de imagen a texto juega un rol importante.

La técnica de PLN en la que se centra este trabajo es la de extracción de información. La extracción de información puede describirse como la tarea de recibir textos no estructurados en lenguaje natural y devolverlos con cierta estructura.

Dentro de extracción de información una aplicación importante es la extracción de eventos, esta consiste en obtener conocimiento específico sobre ciertos incidentes. Puede ser aplicado a eventos del tipo financiero, médicos, climáticos, entre otros.

2.2. Motivación

Inicialmente la motivación principal del proyecto era estudiar el clima y los cambios del mismo, para así luego realizar comparaciones y pronósticos a partir de eventos climáticos acontecidos en el pasado.
En el caso de Uruguay, la forma de obtener los eventos climáticos ocurridos en el pasado es a través de los diarios, que empezaron a publicarse a partir del año 1807. Sin embargo, en esa época, no existían secciones dedicadas exclusivamente a noticias de índole climática.

Si bien la motivación original era enfocarse en el estudio de eventos meteorológicos, durante el transcurso del proyecto, y guiados en parte por la solución elegida, terminamos generalizando el sistema para todo tipo de eventos.

2.3. Objetivo

El objetivo principal de este proyecto es crear una herramienta que permita la extracción de los eventos, su clusterización y etiquetado, de los diarios Uruguayos digitalizados en formato de imagen publicados en la Biblioteca Nacional.

2.4. Resultados esperados

1. Investigar el estado del arte en la extracción de eventos y realizar un resumen de los distintos enfoques.
2. Obtener un proceso de extracción de tuplas $< verbo, argo0, .., argn >$ basados en textos en lenguaje natural del idioma español.
3. Generar un conjunto de clusters etiquetados

2.5. Organización del documento

El documento está organizado de la siguiente forma:

El capítulo 3 está enfocado en brindar algunos conceptos necesarios para el entendimiento del trabajo y el resumen del estado del arte con respecto a extracción de eventos.

En el capítulo 4 se muestra la solución planteada en forma global, explicando la función de cada módulo.

En el capítulo 5 se profundiza sobre el proceso de digitalización del texto (OCR) y la corrección del mismo.

En el capítulo 6 se detalla la extracción de eventos. Se profundiza sobre la solución propuesta, las técnicas utilizadas y las herramientas utilizadas.

En el capítulo 7 se profundiza sobre la clusterización y etiquetado de los clusters, explicando los algoritmos utilizados.

En el capítulo 8 se muestran las evaluaciones realizadas y análisis de los resultados obtenidos.

Finalmente, en el capítulo 9 se detallan los problemas encontrados, conclusiones y lineamientos de trabajos futuros.

3. Marco de trabajo

El capítulo actual tiene como principal objetivo mostrar al lector los trabajos relacionados al proyecto. Así como también una descripción del área en el cual se enmarca el proyecto y los enfoques propuestos en diferentes trabajos.

3.1. Extracción de información

3.1.1. Definición

La extracción de información (IE information extraction) consiste en recuperar información de forma automática de textos no estructurados o semi estructurados y a partir de la extracción generar contenido estructurado. Un sistema de extracción de información encuentra y enlaza la información relevante, mientras ignora la extraña e irrelevante (Cowie J. & Lehnert W., 1996). La extracción de información tomó un papel importante a partir de la década de los ochenta cuando la Agencia de Defensa de los Estados Unidos (DARPA) patrocinó las conferencias MUC (*Message Understanding Conference*) que consistían en una competición entre diferentes grupos donde el objetivo era extraer información relacionada a diferentes temas de interés para el DARPA (véase la Figura 3.2). Para la evaluación de los diferentes grupos fue necesario crear un estándar como por ejemplo precisión, recall y medida-F. El cálculo de estas medidas están definidas en la Figura 3.1

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Figura 3.1

Donde tp representa la cantidad de extracciones correctas, fp la cantidad de extracciones extraídas incorrectamente y fn la cantidad de extracciones que no se extrajeron pero deberían haberse extraído. La medida-F es una relación entre la precisión y el recall, es una forma de combinar estas dos medidas. Alcanza su mejor valor en 1 (precisión y recall perfecta) y el peor en 0.

Conference	*Year*	*Text Source*	*Topic (Domain)*
MUC-1	1987	Mil. reports	Fleet Operations
MUC-2	1989	Mil. reports	Fleet Operations
MUC-3	1991	News reports	Terrorist activities in Latin America
MUC-4	1992	News reports	Terrorist activities in Latin America
MUC-5	1993	News reports	Corporate Joint Ventures, Microelectronic production
MUC-6	1995	News reports	Negotiation of Labor Disputes and Corporate Management Succession
MUC-7	1997	News reports	Airplane crashes, and Rocket/Missile Launches

Figura 3.2

3.1.2. Detección y reconocimiento de eventos

Es una tarea compleja que consiste en múltiples tareas como la detección de entidades con nombre, anchura de eventos, identificación y asignación de argumentos, resolución de correferencia de eventos, entre otros. Consiste en extraer información sobre qué ocurrió, cuándo, dónde y a quién.

Cuando hablamos de reconocer eventos, nos referimos a encontrar una porción de texto con una determinada anchura (una o más palabras), con cero o más argumentos, los cuales pueden ser entidades, tiempos o valores.

Muchos de los artículos explicados luego, utilizan un corpus llamado ACE para entrenar sus algoritmos, por eso es importante aclarar de qué está compuesto dicho corpus.

Corpus ACE

ACE (ACE | Linguistic Data Consortium) es un corpus anotado para desarrollar técnicas de extracción de información de manera automática sobre texto no estructurado. Se basa en la detección y caracterización de entidades, relaciones y eventos. Cabe destacar que tanto las relaciones, eventos y entidades, tienen un tipo y subtipo asociado y que estos son acotados. Inicialmente el corpus estaba en inglés, pero ya se encuentra para otros idiomas, como por ejemplo el español.

3.2. Estado del arte

3.2.1. Enfoque basado en reglas

El enfoque basado en reglas consiste en generar patrones de extracción para recuperar información de los textos. Uno de los principales desafíos en el enfoque de reglas radica en generar reglas lo suficientemente abarcativas y confiables que extraigan la relación de interés.

En el articulo “Automatic Acquisition of Hyponyms from Large Text Corpora” (Hearst, M.A., 1992) utilizan el enfoque de reglas para extraer la relación de hiponimia en textos no estructurados. Parten de generar reglas comunes para la relación para que puedan ser utilizadas sobre diferentes dominios y sin un conocimiento previo del tipo de texto en donde se quiere extraer.

Un ejemplo de regla general que extraiga la relación podría ser:

$SN\ \{, SN\}^*\ y\ otros\ SN$ (Donde SN representa un sintagma nominal)

Por ejemplo para la oración "*Los gatos, perros y otros animales.*" si se utiliza la regla anterior se puede extraer que "*gatos*" y "*perros*" cumple la relación de hiponimia con "*animales*".

El proceso utilizado para la identificación de reglas mostrado en el artículo consiste en lo siguiente:

- Encontrar los patrones que ocurren con frecuencia en diferentes tipos de texto.
- Tomar los que casi siempre indican la relación de interés y puedan reconocerse con poco o ningún conocimiento previo sobre texto.

A los efectos de extraer la mayor cantidad de relaciones, se utiliza la mayor cantidad de reglas posible.

Para poder tener la mayor cantidad de reglas posible surge la idea de generar reglas automáticas a partir de los siguientes pasos:

1. Generar un conjunto de reglas de forma manual a partir de la observación de textos y conocimiento previo de ejemplos de la relación a extraer.
2. Se aplica cada regla en el corpus y como resultado nos devuelve instancias o argumentos de la relación buscada.
3. Para cada tupla de argumentos extraídos en el paso anterior, se busca en el corpus las sentencias que contengan los mismos.
4. Luego se buscan características comunes de las sentencias encontradas y se genera automáticamente la regla que extrae la relación de interés.
5. Cuando se identifica una regla se agrega al conjunto inicial y se vuelve al punto dos.

De todos modos el algoritmo no fue implementado ya que el paso 4 es indeterminado. Con respecto al paso 4, en la tesis "Descubrimiento Automático de Hipónimos a partir de Texto no Estructurado" (Mendoza, R.M.O. and Computacionales, C., 2007) utilizan extracción de secuencias frecuentes maximales (Ahonen-Myka, H., 2002) para la generación de las reglas

automáticas. Los pasos 1, 2 y 3 son iguales al del algoritmo comentado anteriormente, pero en el paso 4 para la generación de nuevas reglas suplantan las palabras de la relación por *etiquetas* (por ejemplo *evento*, *consecuencia*) y generan un conjunto de posibles reglas léxicas utilizando secuencias frecuentes maximales. Las secuencias frecuentes son las secuencias de palabras que aparecen como mínimo una cantidad β (umbral) de veces en los textos analizados. Una secuencia p es una secuencia frecuente maximal si p es frecuente y no existe ninguna secuencia p' en el conjunto de textos tal que p sea una subsecuencia de p' y p' sea frecuente.

Para que la nueva regla se agregue al conjunto de reglas debe cumplir con ciertas características. Que las *etiquetas* estén en la nueva regla a agregar y que no contenga signos de puntuación (esta condición aplica para la relación particular que estaban queriendo extraer). Después de la generación de las reglas, hay que evaluar las mismas y encontrar una medida de confianza con respecto a si extrae correctamente la relación o no. El método utiliza que una regla es más confiable mientras mayor sea la cantidad de tuplas confiables que extraiga y una tupla es más confiable mientras mayor sea la cantidad de reglas que la extraen.

Otros enfoques basados en reglas, están implementados en OpenIE.

OpenIE

La extracción abierta de información (Open Information Extraction en inglés) es la tarea de extraer afirmaciones del texto, sin especificación previa de la relación o del vocabulario (Análisis semántico de lenguaje natural, FIng). Generalmente se extraen ternas (relaciones binarias) compuestas por una relación y dos argumentos. Se denomina abierta debido a que las relaciones no están previamente establecidas, sino que utilizando patrones, las reglas se descubren automáticamente. También se le dice abierta, porque no está pensada para un dominio específico, sino que es para un dominio abierto.

Reverb

Reverb es un algoritmo de OpenIE (Fader et al., 2011), el cual intenta disminuir la cantidad de extracciones incoherentes y las que pierden información relevante respecto de sus predecesores. Para ello, agregan restricciones sintácticas, por ejemplo: las relaciones multi palabras deben comenzar con un verbo y terminar con una preposición y pueden tener una cantidad variable de sustantivos y adjetivos entre medio.

Por otro lado, las restricciones sintácticas pueden hacer que la relación encontrada sea muy específica (al ser esta muy larga), perdiendo así generalidad. Es por esto que se agregan reglas léxicas.

Si bien el algoritmo muestra buenos resultados, en nuestro proyecto fue imposible utilizarlo ya que está hecho para el idioma inglés.

Por otro lado, para el español existen hoy en día dos algoritmos: ExtrHech (Zhila, A. and Gelbukh, A., 2014) y ArgOE (Gamallo, P. and Garcia, M., 2015)

ExtrHech

El procedimiento utilizado por este enfoque, es el de recibir un texto con POS Tagging, aplicar restricciones sintácticas y devolver tripletas $< arg1,\ relación,\ arg2 >$.

Una vez recibido el texto con su POS-TAG, aplica la regla de la Figura 3.2. Este patrón es utilizado para encontrar la relación en la oración. Aquí V representa un verbo con un pronombre reflexivo opcional ó un participio, W puede ser un sustantivo, un adjetivo, un adverbio, un pronombre o un artículo, mientras que P es una preposición seguida opcionalmente por un verbo infinitivo o gerundio. El símbolo * representa cero o más.

$$VREL \rightarrow (V\ W*\ P) \mid (V)$$

Figura 3.3

De aquí en adelante, llamaremos “frase verbal” a todo el matcheo de la regla anterior.

Para encontrar los argumentos de la frase verbal, busca a la izquierda del comienzo de la misma una frase nominal. Esta frase nominal es potencialmente un argumento de la relación. Si encuentra el argumento, busca una frase nominal a la derecha del fin de la frase verbal. Esta nueva frase nominal es el segundo argumento. Para encontrar las frases nominales, utiliza la siguiente expresión regular.

$$NP \rightarrow NP\ (PREP\ NP)?$$

Aquí NP representa un sustantivo con un conjunto de modificadores opcionales como determinantes, adjetivos, números, mientras que $PREP$ es una frase preposicional. El símbolo $?$ representa cero o una vez.

A nivel semántico, el argumento1 representa el agente o el experiente y el argumento2 el objeto directo o circunstancial.

Las reglas recién descritas son las generales, también tiene reglas para casos particulares como participios seguidos por un sustantivo, verbos coordinados por la conjunción, pronombre relativos, entre otras.

ArgOE

Al igual que ExtrHech, ArgOE es un sistema basado en reglas, pero la diferencia con el primero es que éste opera sobre el árbol de dependencia en vez del POS-Tag. Es un sistema que genera tripletas de relaciones, siendo la relación un verbo más un agregado opcional.

Dada una oración con su árbol de dependencia, selecciona todos los verbos que tenga y para cada verbo, escoge los argumentos dependientes de él, cuya función sintáctica sea sujeto, objeto directo, atributo o preposición. Consiste en cinco reglas que combinan lo antes dicho.

Se busca encontrar las proposiciones que sean consideradas coherentes y no sobre-específicas. Para garantizar esto, se definen ciertas reglas: los objetos directos nunca se omiten y las relaciones no pueden contener más de un constituyente.

Dentro de la clasificación anterior se enmarca el articulo ”Unsupervised Techniques for Extracting and Clustering Complex Events in News” (Rusu, D., Hodson, J. and Kimball, A., 2014). El objetivo del artículo es la extracción de eventos (verbo y argumentos) y su clusterización en clases predefinidas, además de extraer las relación de los eventos en una misma oración. Para realizarlo primero identifican el evento basados en el árbol de dependencia y analizan la relación entre diferentes palabras dentro de una misma oración. Para formar el evento extraen el verbo y los argumentos relacionados a este, utilizando REVERB para identificar el sujeto y el objeto, y “Stanford Named Entity Recognizer” para extraer las entidades con nombre y las expresiones temporales. Para relacionar eventos en una misma sentencia toman los verbos que están linkeados directamente, esto les permite aumentar argumentos faltantes en los eventos relacionados. Luego de extraídos los eventos para clusterizarlos utilizan los atributos de los argumentos del mismo (sujeto, objeto y entidades con nombre) para determinar a qué cluster pertenece. La forma de establecer la similitud del evento y el cluster es utilizando diferentes técnicas:

- WordNet[2] super-sense
- BabelNet[3] senses
- BabelNet hypernyms.
- WordNet super-senses, BabelNet senses e hypernyms.

Como se verá en los capítulos 5 y 6, nuestra solución se basa fuertemente en el artículo descrito.

[2] https://wordnet.princeton.edu/
[3] https://babelnet.org/

3.2.2. Enfoque basado en clasificación

En “The stages of event extraction.” (Ahn, D., 2006) se trata la extracción de eventos como un pipeline de 4 etapas. Cabe mencionar, que en este artículo no se trabaja en el reconocimiento de entidades con nombre, sino que este ya se asume como sabido. Cada etapa es tratada como un problema de clasificación de manera independiente del resto (al ser un pipeline dependen únicamente de la etapa previa). Los clasificadores utilizados son: TiMBL, un clasificador basado en memoria y MegaM, un clasificador de máxima entropía, el cual no asume independencia entre los features.

En la etapa 1, se identifica la anchura de los eventos. Se basa en detectar de donde a donde va el evento en el texto. Para esta etapa, sólo se consideran eventos cuyo largo máximo es de una palabra. Los eventos tienen que pertenecer a un grupo acotado de categorías sintácticas. Por lo tanto, la etapa 1 consiste en clasificar cada palabra del documento, en uno de los 34 tipos de evento (incluido None, que no es evento).

Como la mayoría de las palabras no son eventos, el conjunto de entrenamiento tiene mucho sesgo con la clase None, por lo que primero se aplica un clasificador binario y luego un clasificador multiclase sobre las instancias positivas. Los features utilizados para la clasificación son:

- features léxicos[4]
- features de wordnet
- contexto de la palabra (3 palabras hacia atrás/adelante)
- features del árbol de dependencia.

En la etapa 2, se identifican los argumentos del evento. Para ello, se consideran todas las entidades, tiempos y valores que ocurren en la oración y se emparejan con el evento con el fin de transformarlo en una tarea de

[4] features léxicos se refiere sin ser exhaustivo a características como: lema, palabra textual, palabra textual en minúsculas, pos tag, profundidad de en el árbol de dependencia.

clasificación. Hay que tener en cuenta que los eventos soportan determinados roles.

Se prueba con dos métodos de clasificación: un único clasificador para todo y un clasificador por tipo de evento.

Los features que se tienen en cuenta son:

- features léxicos
- tipo de evento
- mismas features que el evento, pero a la entidad
- determinante de la entidad
- camino del árbol de dependencia desde el evento a la entidad, expresado como las palabras con su pos tag

En la etapa 3, se habla sobre los atributos de cada evento, por ejemplo si es positivo o negativo, su tiempo verbal, entre otros. Decidimos ignorar esta etapa ya que no nos interesa clasificar este tipo de atributos para nuestro proyecto.

En la etapa 4 se resuelve la correferencia de eventos. Se utiliza un clasificador probabilístico. Para cada evento que aparece en el documento, se clasifica utilizando cada uno de los eventos anteriores a él, para ver si hacen referencia al mismo cluster[5] o no. Llamamos candidato a cualquiera de los eventos anteriores y anáfora al evento actual que se quiere clasificar. Los features utilizados son:

- anchura del candidato y anáfora, con su pos tag y minúsculas.
- tipos de evento de cada uno.
- profundidad en el árbol de dependencia de cada uno.
- distancia en sentencias entre candidatos.
- números, heads y roles de argumentos compartidos.
- número, heads y roles que no son argumentos de la anáfora y del candidato.

[5] Cluster hace referencia a conjunto de eventos relacionados.

A continuación vamos a hablar sobre la extracción conjunta de eventos y entidades en el contexto de un documento basado en el trabajo de Yang y Mitchell (Yang, B. and Mitchell, T., 2016, Joint Extraction of Events and Entities within a Document Context.).

La extracción de eventos y entidades suele hacerse a nivel de sentencias. El objetivo de este enfoque es tener acceso a información del contexto a nivel del documento para realizar predicciones sensibles al contexto.

Además, la extracción de eventos y entidades suele hacerse mediante un pipeline secuencial de etapas individuales, en donde la extracción de eventos y entidades se realiza de forma individual en distintas etapas mediante clasificadores locales independientes.

Un problema que tiene este enfoque es que los errores de una etapa son propagados a las etapas siguientes a través del pipeline. Por ejemplo en la frase: "Una tormenta derribó dos casas en Montevideo." si la palabra 'Montevideo' es clasificada erróneamente como una entidad de tipo PERSONA, el sistema no va a extraer a 'Montevideo' como el lugar en donde ocurrió la tormenta.

Otra limitante de la mano de este problema es que no se puede obtener feedback de etapas posteriores para realizar la tarea de la etapa actual.

Para superar estos problemas es que surge la extracción conjunta de eventos y entidades.

El aprendizaje se divide en tres tareas:

- Aprendizaje de la estructura interna de los eventos
- Aprendizaje de las relaciones entre eventos
- Aprendizaje para la extracción de entidades

Dado un documento x, primero se determinan los candidatos a eventos T y los candidatos a entidades N[6]. A cada candidato $i \in T$ lo asociamos con una variable discreta t_i que toma valores de nuestra lista de tipos de eventos más

[6] Una forma de obtener los candidatos a eventos es quedarse con las k-mejores predicciones de un modelo CRF (Conditional random fields) para eventos y lo mismo para un modelo CRF para entidades.

un tipo $NONE$ que indica que es de otro tipo de evento que no está en nuestra lista o que no es un evento. Denotamos N_i al conjunto de candidatos a entidades que son potenciales argumentos del candidato a evento i, y a cada $j \in N_i$ le asociamos una variable discreta r_{ij} que representa la relación evento-argumento entre el candidato a evento i y el candidato a entidad j. r_{ij} toma valores de nuestra lista de roles semánticos más un valor $NONE$ que indica roles inválidos.

A su vez, a cada candidato a argumento j se le asocia una variable discreta a_j que toma valores de nuestra lista de tipos de entidades (PERSONA, ORGANIZACIÓN, LUGAR, etc.) más un valor $NONE$ que indica tipos de entidades inválidos.

Ahora que tenemos nuestras variables discretas (t_i, r_{ij}, a_j) y nuestras observaciones (i, N_i, x), formulamos modelos probabilísticos para aprender cada una de las tareas de aprendizaje.

Aprendizaje de la estructura interna de los eventos:

Se usa un modelo probabilístico para las variables (t_i, r_{ij}, a_j) condicionado a las observaciones (i, N_i, x) basada en funciones de features (se pueden ver los features usados usualmente en la Tabla 3.1) y con parámetro θ a estimar durante el entrenamiento. Durante el entrenamiento hallamos el θ óptimo usando el método de máxima verosimilitud con regularización L2. Se usa el algoritmo L-BFGS para optimizar el objetivo de entrenamiento.

Aprendizaje de las relaciones entre eventos:

Se usa un modelo probabilístico para las variables (t_i, t_i') para cada par de candidatos a eventos (i, i')[7] condicionada a las observaciones (i, i', x) basada en

[7] Se consideran los candidatos a eventos que aparezcan en la misma frase, o que estén conectados por un sujeto/objeto correferente si están en distintas frases. Esta etapa es fundamental para que la extracción sea sensible al contexto del documento.

funciones de features y con parámetro ϕ a estimar durante el entrenamiento. Se usan features específicos a los eventos (los mismos de la Tabla 3.1) así como features relacionales entre los eventos.
Algunos de estos son:

1. Están conectados por una relación de conjunción de dependencia (basado en un parsing de dependencia).
2. Comparten un sujeto o un objeto (basado en parsing de dependencia y resolución de correferencias).
3. Tienen el mismo lema central.
4. Comparten un frame semántico basado en FrameNet[8].

Se usa L-BFGS para computar máxima verosimilitud para ϕ durante entrenamiento.

Aprendizaje para la extracción de entidades:

Para extracción de entidades se entrena un modelo CRF (el mismo usado para determinar los candidatos) basado en el esquema BIO. Se utilizan features comunes en este tipo de tarea como por ejemplo:

1. Palabra actual y su part-of-speech tag.
2. Palabras del contexto en una ventana de tamaño dos.
3. Lo que se suele llamar "word type" que refiere a features como si empieza con mayúscula o si son todos dígitos, etc.

[8] https://framenet.icsi.berkeley.edu/fndrupal/

Category	Type	Features
Trigger	*Lexical resources:* WordNet Nomlex FrameNet Word2Vec	1. lemmas of the words in the trigger mention 2. nominalization of the words based on Nomlex (Macleod et al., 1998) 3. context words within a window of size 2 4. similarity features between the head word and a list of trigger seeds based on WordNet (Bronstein et al., 2015) 5. semantic frames that associate with the head word and its p-o-s tag based on FrameNet (Li et al., 2014) 6. pre-trained vector for the head word (Mikolov et al., 2013)
	Syntactic resources: Stanford parser	7. dependency edges involving the head word, both lexicalized and unlexicalized 8. whether the head word is a pronoun
Argument	*Lexical resources:* WordNet	1. lemmas of the words in the entity mention 2. lemmas of the words in the trigger mention 3. words between the entity mention and the trigger mention
	Syntactic resources: Stanford parser	4. the relative position of the entity mention to the trigger mention (before, after, or contain) 5. whether the entity mention and the trigger mention are in the same clause 6. the shortest dependency paths between the entity mention and the trigger mention
Entity	*Entity resources:* Stanford NER NELL KB	1. Gender and animacy attributes of the entity mention 2. Stanford NER type for the entity mention 3. Semantic type for the entity mention based on the NELL knowledge base (Mitchell et al., 2015) 4. Predicted entity type and confidence score for the entity mention output by the entity extractor described in Section 3.3

Tabla 3.1 (Bishan Yang & Tom Mitchell, 2016)

Inferencia Conjunta:

Para lograr la inferencia conjunta se plantea un objetivo de optimización que permite el flujo de información entre los tres modelos locales y que encuentra una asignación óptima global para nuestras variables. En particular a nuestras variables de tipo de evento t, variables de rol de argumento r, y variables de tipo de entidad a.

Este objetivo se puede formular como un problema de ILP(Programación Lineal Entera). Una forma de resolver el problema es buscando la solución a la relajación del problema usando un algoritmo de descomposición dual como AD^3.

4. Solución propuesta

Como se aprecia en la Figura 4.1, el sistema recibe imágenes de diarios y se entregan al módulo OCR. Este transforma la imagen en texto, el cual cuenta con varios errores ortográficos (errores debido a la mutación del lenguaje o generados por el proceso de OCR). Por ésto, es necesario el módulo “Corrector” como método de postprocesamiento del OCR. Una vez corregido el texto, se realiza la extracción de eventos sobre los mismos. Con las tuplas de los eventos extraídos, se realiza la clusterización de los mismos según su afinidad semántica. Finalmente, ya con los clusters generados, se procede a etiquetarlos en base a categorías o secciones utilizada por diarios actuales.
La arquitectura es la siguiente:

Los textos de entrada corresponden a prensa extraída de la Biblioteca Nacional del Uruguay del siglo XIX. Estos textos son imágenes escaneadas de los diarios originales por lo que presentan deterioro; conteniendo manchas, dobleces, además del propio envejecimiento del papel. También tienen errores generados por el escaneo como desalineamiento del texto.
Al estar los diarios en formato de imagen, tuvimos que usar un OCR para transformarlos a texto digitalizado.

Debido a la mala calidad de la salida del OCR y el lenguaje de la época, tuvimos que implementar un módulo de corrección ortográfica. El mismo utiliza frecuencia de n-gramas, distancia de Levenshtein, para corregir los errores de la mejor forma posible.

Una vez que los textos están corregidos procedemos a la extracción de los eventos. Basados en el estado del arte, existen diferentes métodos:

- Aprendizaje automático supervisado
- Aprendizaje automático no supervisado
- Basado en reglas
- Híbridos

La falta de textos anotados hizo inadecuado el uso de métodos supervisados. La poca cantidad de datos hizo inviable los métodos basados en aprendizaje automático. Por lo que se optó por un sistema basado en reglas.

Para extraer los eventos se analiza la oración con su información lingüística y se generan tuplas de la forma $< verbo, arg0, arg1, ..., argn >$. Está fuera del alcance de este proyecto los eventos que se extienden a más de una oración.

Una vez procesados los textos y extraídos los eventos, pasamos a agruparlos en clusters según la similitud semántica utilizando la técnica de word embedding.

Como etapa final, se busca etiquetar los clusters. Para ello, se aprovecha la división en secciones que cuenta hoy en día la prensa digital uruguaya. La idea es asignarle una sección a cada cluster.

En resumen el sistema consta de cuatro módulos importantes:

- Módulo preprocesamiento
    - OCR
        - Entrada: imágenes escaneadas de los diarios
        - Salida: texto digitalizado
    - Corrector
        - Entrada: texto digitalizado
        - Salida: texto corregido
- Módulo extracción de eventos

    - Entrada: texto corregido
    - Salida: lista de tuplas $[< verbo, arg0, ..., argn >]$
- Módulo clusterización
    - Entrada: lista de tuplas $[< verbo, arg0, ..., argn >]$
    - Salida: clusters de eventos similares
- Módulo etiquetado de clusters
    - Entrada: clusters de eventos similares
    - Salida: clusters etiquetados

En los siguientes capítulos se describirán en detalle todos los módulos.
A modo de ejemplo a continuación se detalla el procesamiento aplicado sobre ciertas oraciones de un diario.
Al realizar el OCR sobre las oraciones del diario, el resultado devuelto por la herramienta Abby Cloud es:

Otro querer de la nación, bien pronun-
ciado, es vivir bajo el réjimen constitucio-
nal. El se ha manifestado vivo y enérjico
en todas las ocasiones en que se ha presen-
tado un semblante de paz.
Pero querer la independencia es querer el ejer-
cicio franco y completo de la Soberanía.
Aprovechemos la ocasión de im-
ponerle una ley que en otro caso no acep-
taría de ningún modo.

Al aplicarle el corrector el resultado se traduce a:

Otro querer de la nación, bien pronunciado
es vivir bajo el régimen constitucional.
El se ha manifestado vivo y enérgico

en todas las ocasiones en que se ha presentado
un semblante de paz.
Pero querer la independencia es querer el ejercicio
franco y completo de la Soberanía.
Aprovechemos la ocasión de imponerle
una ley que en otro caso no aceptaría
de ningún modo.

En la Figura 4.2 se muestran las diferencias al aplicarle el corrector. Como se puede ver el corrector corrige:

- Palabras que se encuentran separadas por guión al final del margen.
- Palabras antiguas que antes se escribían con *j* y que hoy se escriben con *g*.

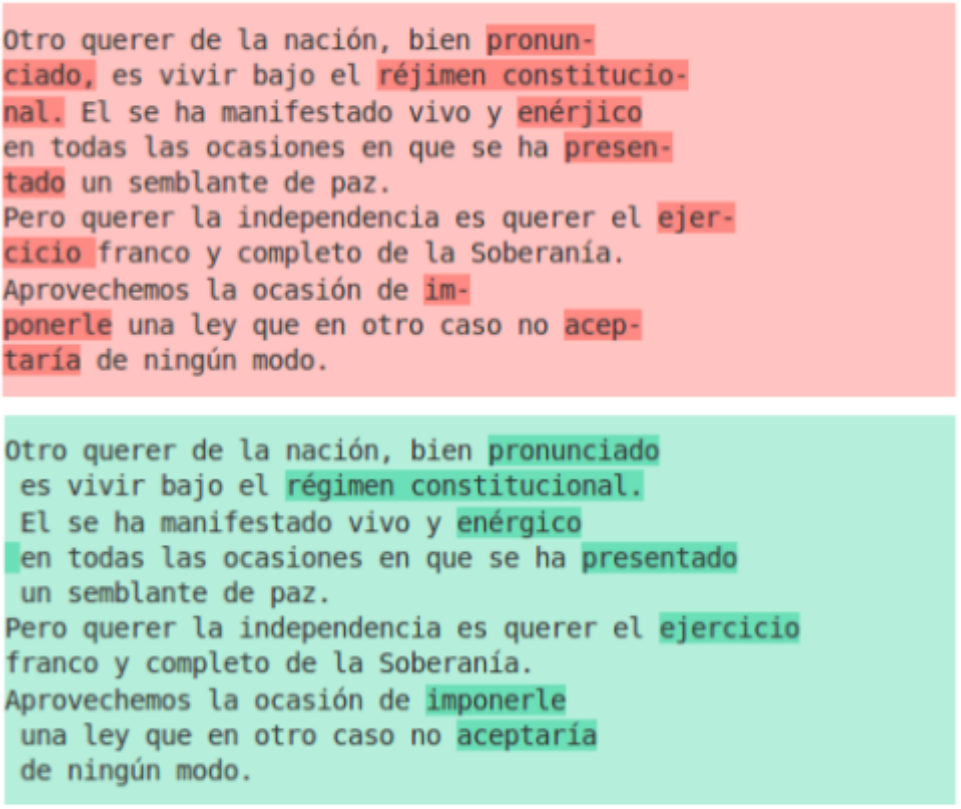


Figura 4.2 Diferencias entre salida OCR y texto corregido

Luego de intentar corregir los errores de ortografía, se extraen los eventos. El resultado de la extracción se muestra a continuación:

< es, Otro querer de la nación bien pronunciado, vivir bajo el régimen constitucional >

< se ha manifestado, , vivo y enérgico >

En este caso la oración original es: *"El se ha manifestado vivo y enérgico..."*, pero debido a que *"El"* no llevaba tilde, el analizador de dependencias no lo detectó como un sujeto, sino como un especificador (determinante), por eso es que el evento no tiene sujeto.

< se ha presentado, un semblante de paz, en todas las ocasiones >

Aquí se puede apreciar cómo sustituye el pronombre relativo *"que"* por *"todas las ocasiones"*.

< es, Pero querer la independencia, querer el ejercicio franco y completo de la Soberanía >

< aprovechemos, , la ocasión de imponerle una ley que en otro caso no aceptaría de ningún modo >

Finalmente se clusterizan los eventos extraídos y se le asigna una etiqueta a cada cluster. La etiqueta p}uesta para todos los eventos recién mencionados fue de ***política.***

5. Obtención y preprocesamiento de noticias

Como primer paso se necesita de los textos a los cuales realizarle la extracción de información. Por esto, se propone la creación de un módulo para descargar los diarios de la BNU (Biblioteca Nacional del Uruguay). Una vez descargados los diarios el siguiente punto es la conversión de las imágenes de los diarios en texto no estructurado. En el contexto de este proyecto este último punto es sumamente importante para poder manipular los textos de los diarios y de esta forma poder extraer los datos relacionados a eventos. Lo que es claro es que mientras mejor sea la salida del OCR, mejor será la extracción de información sobre la misma.

5.1. Web Scraper

Web scraping es una técnica utilizada mediante programas de software para extraer información de sitios web. Usualmente, estos programas simulan la navegación de un humano en la Web ya sea utilizando el protocolo HTTP manualmente, o incrustando un navegador en una aplicación.
Se enfoca principalmente en la transformación de datos sin estructura en la web(como el formato HTML) a datos estructurados que pueden ser almacenados y analizados en una base de datos, en una hoja de cálculo o en alguna otra fuente de almacenamiento.
En sí de Web Scraping, utilizamos un subconjunto de técnicas acotado. Obtenemos los archivos en formato PDF, desde una ruta relativa a la web de la BNU que contiene las publicaciones de los diarios digitalizados del siglo XIX (http://bibliotecadigital.bibna.gub.uy:8080/jspui/handle/123456789/18).
A la hora de seleccionar la librería, estuvimos entre las dos más utilizadas en Java. Estas librerías son JSoup y Jaunt. Dado que no hay requerimientos

funcionales ni no funcionales importantes sobre este módulo y debido a que ambos son muy fáciles de utilizar, nuestro criterio de selección fue simplemente el hecho de que JSoup es open source y Jaunt no.

5.2. OCR de imagen a texto

Reconocimiento Óptico de Caracteres, u OCR (por sus siglas en inglés Optical Character Recognition), es una tecnología que permite convertir diferentes tipos de documentos, tales como documentos en papel escaneados, archivos de PDF o imágenes captadas por una cámara digital en datos editables y con opción de búsqueda (Abby OCR.).
Los diarios digitalizados de la BNU son simplemente imágenes en formato PDF. Veamos entonces las posibles optimizaciones que se le puede hacer al OCR para mejorar la salida. En esta sección veremos algunas de las técnicas más usadas, así como mencionaremos trabajos relacionados que han utilizado estas técnicas. Para mejorar la salida del OCR se pueden aplicar diferentes técnicas en diferentes puntos del procesamiento: antes, durante y después; y utilizar una amplia variedad de recursos: diccionarios, desambiguación lingüística, etiquetado gramatical, etc. Debido a esto, aquí vamos a dar un resumen general de algunos de los diferentes enfoques disponibles.

El siguiente resumen está basado fuertemente en el trabajo “Corrección automática de errores de OCR en documentos semi-estructurados” (Pablo A. Paliza,2016)

5.2.1. Preparación de la imagen

El procesamiento de imágenes puede contribuir en gran medida para mejorar la exactitud del OCR, por ejemplo, mejorando la iluminación no uniforme de las imágenes. Este método es muy útil cuando las imágenes a procesar están deterioradas, ya que al mejorar la calidad de la imagen el motor de OCR puede identificar mejor el texto de la imagen.
Otro enfoque que se puede realizar antes de procesar la imagen con el OCR es la identificación de texto manuscrito en partes de la imagen, ya que en general este tipo de texto no es bien manejado por los programas de OCR. En nuestro caso particular este enfoque no es necesario, ya que los diarios no contienen texto manuscrito.
Exploramos utilizar en nuestra solución la herramienta ImageMagick[9], la cual es muy recomendada para el tratamiento de imágenes. Conseguimos buenos resultados analizando imágen por imágen (o diario por diario), pero la terminamos descartando ya que no existía una configuración general que sea la óptima para todas las imágenes. Por lo que analizar cada imagen, era inviable.

5.2.2. Durante el OCR

Si se puede acceder a la programación interna del software de OCR, puede ser posible incrementar la exactitud durante el proceso de reconocimiento. Los motores de OCR de código abierto brindan la posibilidad de estudiar cómo funcionan estos programas y se pueden realizar modificaciones para que se adecúen al corpus que se quiere procesar. Por ejemplo, adaptándolo para alfabetos no estándares o diferentes tipos de fuentes de texto. También es posible entrenar estos motores con textos similares o una fracción del corpus

[9] https://www.imagemagick.org/script/index.php

que se quiere procesar. Un ejemplo de esto es reportado en “Report on the comparison of tesseract and abbyy finereader ocr engines” (Heliński et al., 2012), donde un sistema de código abierto es específicamente entrenado y ajustado para reconocer textos históricos escritos en lenguas bereberes transcritos en latín. Otra forma de corrección de palabras, es la de utilizar herramientas de OCR que tengan la característica de marcar un nivel de confianza binaria a nivel de caracteres, aprovechando esta información para hacer los ajustes necesarios.

Este enfoque no fue utilizado por nosotros para este trabajo, en particular por dos razones: la primera es que modificar la herramienta de OCR para que se adecúe a nuestro contexto puede llevar un trabajo considerable lo cual escaparía al alcance del proyecto, y la segunda es que en general (y como se verá en más detalle en la sección 5.2.4) las herramientas de código abierto suelen tener resultados bastante peores que las herramientas que no son de código abierto. Es por esto también que se suele optar por enfoques que mejoran la salida del OCR a posteriori.

5.2.3. Después del OCR

Una manera de corregir los errores de OCR a posteriori es utilizando el modelo del canal ruidoso (noisy channel model) (Jurafsky, D. and Martin, J., 2018), en el cual un mensaje es enviado (texto del diario) y distorsionado debido al ruido en el canal de comunicación (el proceso de OCR) y el receptor intenta decodificar o entender qué fue enviado.

Otra línea de trabajo, es utilizar la métrica de distancias de edición (comúnmente la distancia de Levenshtein o alguna de sus variaciones) entre las palabras obtenidas por el OCR y las palabras presentes en un diccionario.

Alternativamente las salidas de varios motores de OCR pueden ser comparadas utilizando un sistema de votación para seleccionar o combinar la mejor salida.

Cada uno de estos enfoques tienen ventajas y desventajas que fueron evaluadas en vista a las colecciones de texto que se esperan corregir.

Distancia de Levenshtein

Para medir la distancia entre dos cadenas de caracteres se puede utilizar la distancia de Levenshtein. Esta consiste en el número mínimo de operaciones a aplicar a una de las cadenas para que se transforme en la otra. Dónde operación puede ser una sustitución, eliminación o inserción de un carácter.

Distancia de Levenshtein con peso

La distancia de Levenshtein explicada anteriormente asigna el mismo peso para cada operación, independientemente del tipo de operación y letras afectadas. En el caso de la distancia de Levenshtein asignando un peso a las operaciones consiste en asignar un valor diferente de acuerdo al cambio realizado, como por ejemplo que cueste la eliminación más que la inserción, o más profundamente que para ciertos casos sea más costoso cambiar un carácter “o” por “x” que un carácter “o” por “c”. Ya que para un procesador de OCR es más fácil confundir una “o” por “c” por ser similares los gráficos de las letras. Otra de las opciones es el modelo del canal ruidoso, en donde se asume que el error pudo ser producido por alguna de las operaciones de inserción, eliminación, sustitución y transposición. Otra opción para corregir textos escritos en teclado puede ser asignarle menor costo a las letras que se encuentran a menor distancia física en el teclado de las que se encuentran más distanciadas.

Modelos de Lenguaje: N-Gramas

Un modelo del lenguaje estadístico asigna una probabilidad a una secuencia de m palabras $P (w_1 , \dots , w_m)$ mediante una distribución de probabilidad. El objetivo de este modelo es calcular la probabilidad de una frase. Tener una forma de estimar la verosimilitud de diferentes frases es útil en muchas aplicaciones de procesamiento de lenguaje natural. Veamos algunos ejemplos:

Traducción automática:
P("glóbulos rojos") > P("glóbulos colorados")

Corrección ortográfica automática:
El mercado está a diez minutos de mi casa.
P("El mercado está a diez minutos de mi casa.") > P("El mercado está a diez minutas de mi casa.")

Reconocimiento del habla:
P(“vientos fuertes”) > P(“bien, tos fuerte”)

En nuestro caso lo utilizaremos para corregir errores que haya cometido el OCR, como también errores presentes en el diario.

La probabilidad de una frase F se calcula como $P(F)=P(p_1,p_2,p_3,..)$ donde p_i son las palabras en la frase F. Usando propiedades de probabilidad tenemos que $P(F)=(p_1,p_2,p_3,..)=P(p_1)*P(p_2|p_1)*P(p_3|p_1,p_2)...$

Veamos un ejemplo con la frase “el viento sopla”:

$P(el,viento,sopla) = P(el)*P(viento|el)*P(sopla|el,viento)$

Para calcular esto necesitamos un corpus en donde calcular las probabilidades de los N-gramas para cada N; la probabilidad del unigrama “el” (es simplemente la cantidad de veces que la palabra aparece en el corpus), la probabilidad del bigrama “el viento” (cantidad de veces que aparece el bigrama sobre la cantidad de veces que aparece el unigrama “el”), y la probabilidad del trigrama “el viento sopla” (que se calcula como la cantidad de veces que aparece el trigrama sobre la cantidad de veces que aparece el

bigrama “el viento”), y así sucesivamente en caso de que la frase sea más larga.

Esta probabilidad permite predecir la palabra siguiente más probable lo cual nos será útil en el proceso de corrección de los diarios.

5.2.4. Evaluación de herramientas

Hay una gran variedad de herramientas OCR (Comparison of optical character recognition software - Wikipedia.). Entre tantas herramientas, hubo que elegir un criterio de filtrado para luego realizar las pruebas con un conjunto más reducido. Ese criterio se basó en tres requisitos: el primero era que no esté descontinuado, para ello exigimos que la última actualización haya sido posterior a 2012; el segundo requisito era que tenga soporte para el español, un criterio más que fundamental para nuestro proyecto debido a que todos los diarios se encuentran en español. El tercer requisito fue que como mínimo tenga una versión de prueba. Finalmente, con las herramientas resultantes evaluamos el resultado de la versión de prueba y el costo de la versión paga para un mínimo de 4000 páginas.
Al aplicar el primer filtro nos quedamos con estas 14 herramientas: ABBY CLOUD[10], Asprise OCR SDK[11], Dynamsoft OCR SDK[12], GOCR[13], LEADTOOLS[14],

[10] https://www.ocrsdk.com/

[11] https://asprise.com/royalty-free-library/java-ocr-api-overview.html
[12] https://www.dynamsoft.com/Products/.net-ocr-component.aspx
[13] http://jocr.sourceforge.net/
[14] https://www.leadtools.com/

MeOCR[15], Nicomsoft OCR SDK[16], OCR.net[17], OCR.space[18], OmniPage[19], PDF OCR X [20], Screenworm[21], Tesseract[22], Yunmai OCR SDK[23].

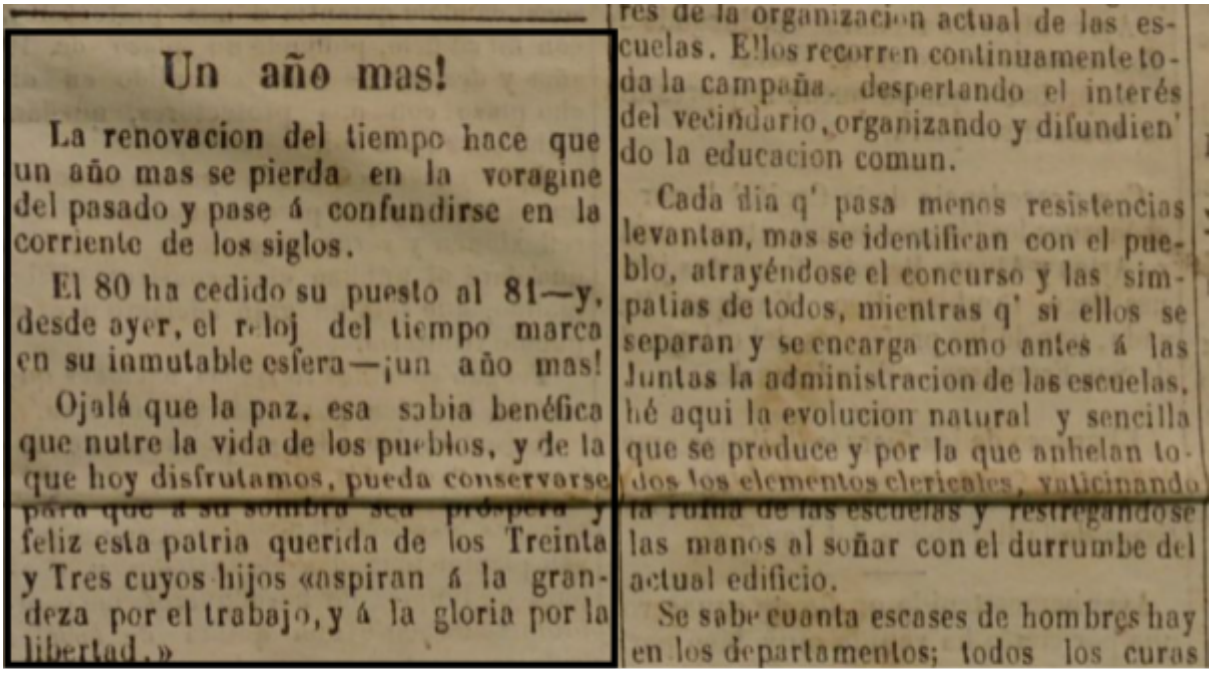
Luego al exigirles que soportaran español, excluimos solamente a Screenworm, quedándonos con las restantes 13 herramientas.

Al aplicar el último filtro nos quedamos con las siguientes 10 herramientas: ABBY CLOUD, Asprise OCR SDK, Dynamsoft OCR SDK, LEADTOOLS, MeOCR, Nicomsoft OCR SDK, OCR.net, PDF OCR X, Tesseract, Yunmai OCR SDK.

A partir del resultado anterior probamos las diferentes herramientas sobre varios diarios para realizar la comparación de las mismas.

Mostraremos el resultado de las diferentes herramientas sobre el siguiente párrafo (marcado con el recuadro negro) del diario “LaTrinidad403” (Figura 5.1):

[15] http://www.meocr.com/
[16] https://www.nicomsoft.com/
[17] http://ocr.net/
[18] https://ocr.space/
[19] https://www.nuance.com/en-gb/print-capture-and-pdf-solutions/optical-character-recognition/omnipage.html
[20] https://solutions.weblite.ca/pdfocrx/index.php
[21] http://www.screenworm.de/
[22] https://github.com/tesseract-ocr/tesseract
[23] http://www.yunmai.com/en/home.html



Un año mas!

La renovacion del tiempo hace que un año mas se pierda en la voragine del pasado y pase á confundirse en la corriente de los siglos.

El 80 ha cedido su puesto al 81—y, desde ayer, el reloj del tiempo marca en su inmutable esfera—¡un año mas!

Ojalá que la paz, esa sabia benéfica que nutre la vida de los pueblos, y de la que hoy disfrutamos, pueda conservarse para que á su sombra sea próspera y feliz esta patria querida de los Treinta y Tres cuyos hijos «aspiran á la grandeza por el trabajo, y á la gloria por la libertad.»

res de la organizacion actual de las escuelas. Ellos recorren continuamente toda la campaña, despertando el interés del vecindario, organizando y difundiendo la educacion comun.

Cada dia q' pasa menos resistencias levantan, mas se identifican con el pueblo, atrayéndose el concurso y las simpatias de todos, mientras q' si ellos se separan y se encarga como antes á las Juntas la administracion de las escuelas, hé aqui la evolucion natural y sencilla que se produce y por la que anhelan todos los elementos clericales, vaticinando la ruina de las escuelas y restregándose las manos al soñar con el durrumbe del actual edificio.

Se sabe cuanta escases de hombres hay en los departamentos; todos los curas

Figura 5.1

LEADTOOLS
Un año mas! cuelas. Ellos recurrí n continuamente to da la campaña, despertando el interés Domingo 2?S. Isidoro obispo y mtr. La renovación del tiempo hace que del vecindario,organizando y difundien Lunes 3?Sta. Gcnoveba virgen, S. un año mas se pierda en la vorágine do la educación común. Florencio obispo. del pasado y pase á confundirse en la Cada rlia q' pasa menos resistencias Martes 4?Sis. Gregorio, Aquilino corriente de los siglos. levantan, mas se identifican con el pue y compañeros mírlir. El 80 ha cedido su puesto al 81?y, blo, atrayéndose el concurso y las sim Miércoles 5?Sis. Telésforo papa y desde ayer, el relej del tiempo marca palias de todos, mientras q' si ellos se mártir y Estefanía virgen.

en su inmutable esfera?¡un año mas! separan y se encarga como antes á las L? cuestión portadas ha sido ya re
Juntas la administración de las escuelas suelta satisfactoriamente.
Ojalá que la paz, esa sabia benéfica lié aqui la evolución natural y sencilla En el próximo número indicaremos
que nutre la vida de los pueblos, y de la que se produce y por IB que anhelan lo- con detalle los caminos y pasos que han
que hoy disfrutamos, puedJaa ccooiunnsseerrvvaarrssee.! Oos loa eWmenVos clerical mam Mali " ?????o . i U Wi.ntjm>j
prv.li II .111 ilHUrJlU jui p Mptiu1 j llll IIB l.ls ft.l'IIMIHs' y
feliz esta patria querida de los Treinta las manos al soñar con el durrumbe del Hemos recibido el importante alma
y Tres cuyos hijos «nspiran á la gran actual edificio. naque que anualmente distribuye a sus
deza por el trabajo, y á la gloria por la Se sube cuanta escases de hombres hay favorecedores «El Siglo.»
libertud.»

Tabla 5.1

Notamos que la herramienta LeadTools (Tabla 5.1) no hace un uso correcto del “layout analysis”. Es decir, no segmenta correctamente el texto, en este caso las columnas del diario. Para este proyecto es necesario que la herramienta pueda leer por zonas como columnas, ya que los diarios a analizar en su mayoría poseen tres columnas en donde se muestran las noticias y si no funciona correctamente entonces los enunciados pierden el sentido.

MeOCR

POUISIIO ; P&#65533;. l' ' ; l'' I .1

(&#65533;ir(&#65533;liumd

4 ut I 'a' vs s. A w 's' &#65533; ra

mm t lar na &#65533; JUSTO P. OSTIX

rnr s KlmontrnlohaU g do.bomuroiid I &#65533; Lss ameoocss rl m u ha

Uikfwlmf~ntw Inoceaciu papa d. Iw Iorrlns u. sa recuuden dal. Jm.

J Vrclot', &#65533;hr ven trece nolrciasr puesto d C Prrecla,

&#65533; Ua si lo numbrmlo ahogarlo dc- b lln d Juml

frnvor ds ln Jacto B. Johlafo el Or. U. &#65533; algunos fuerces gua re lraUsa al

pggggQQQ Jos&#65533;B. Uarr&#65533;loxn ' notle del Ulo 'Jegro &#65533; rin trsslndadan

Ks bise el rumor aun corci&#65533; nueva- nbreor s las fronterw,dq, Baldeando,

uarsmsn,rsu o ac uatsso. melle de lawenunciadsl Mtnittto Uuc Cerro Largo y Tuou re nb&#65533;

K&#65533;chcn.

sur ss nru unicuci nea informsndole dd

Lfpmm6 Ul IUOfileUIO &#65533; Una curia d Corrienms dice que I

ilemp o ' qr" 'eom' vi i r srr. altos wunmete J elsegun&#65533;uur tomar nin &#65533;hi m ln evhs dos localidadesl L m

Tabla 5.2

Al probar la herramienta meOCR (Tabla 5.2) notamos que las zonas detectadas no coinciden con las zonas del diario, además de que para las zonas detectadas el resultado no tiene la calidad mínima aceptada por nosotros ya que la mayoría de las palabras no las reconoce correctamente. Por lo que esta herramienta queda descartada.

Nicomsoft OCR SDK
Un ano mas!da la campaáa. despertando el interés ijomlngo 2—S. kidoro obispo y mEr. , · . . del veCiñdurio ,gtganizando y difundieo' Lunes 3—Sla. Genoveba vtrgen, & La rencmcion del tiempo hace que do la educación común, Flotencio obispo. un alío mas se pierd& v;ñ la voragine . , , , _ 'Mules 4—SIS. Gre6orio, Aquilino del pasado y pose a confundirse en 18 CBda día q posa menos resMemcíaS y compnñeros mártir. comente de Ion siglos. levonlan, mas se idenlifirtín con el ,rme¶,. Miércoles 5— S18. Telésforo papa y Md, olrayéndose e) concurso y las sim" ñijártir v Eslefania virgen. ·' El 80 ha cedido su pupg|o al 8l_yv pMias de lodos, mientras q° si ellos se '"

desde ayer, el tó lnj del tiempo rnarco separan y se'encar€a como onteS a las L? cuestión portadas ha sido ya re-f .
ci) su inmutable esfera—¡un a Do mas! Juntos la aduiinistracion de 106 egcuelas. suelta galisfaclor]amenle.
Ojnl4 que la pnz. esa sjbia benéfica lié aquí la evolución naiural y sencilla
En el próximo Umero indicaremos
·que nutre la vida de los puc'bbs, y de la que se produce y por Ib que Rnhe|nn 10· , cün de\a\\e los Ctjl])il](js y pasos que han
, " que 1)0 dishulamo% ueda conservarsc t)O \ e\emen1o e\ 'ríe
..
.
feliz esta patria querida de os Treinta las mnc's o] soñar con el durrurnbe del
Hemos recibido el importante alma-
y Tres cuyos·hijns «nspiran ti la gran- adual edificio. naque qcie anualmente ásiribuye a sus
deza por el trebojmy á 1"a gloria porlo Se $obp éuonta escoses de hombrp hay fuvoreeedores «El Siglo. ·
Jibertud .))

Tabla 5.3

Para el caso de la herramienta Nicomsoft OCR SDK (Tabla 5.3), la salida del procesamiento da como resultado un texto respetando la estructura original (columnas del diario), cuando en realidad para poder procesarlo necesitamos un formato de texto sin estructura. Además del formato de salida, el resultado con respecto al texto original es de muy baja calidad.

Con respecto a la herramienta Yunmai OCR SDK el problema fue que no pudimos acceder a la versión de prueba porque estaba solo accesible para usuarios de países asiáticos.

En el caso de OCR.net, la aplicación es de escritorio y no cuenta con una api para poder integrarse, este criterio es fundamental para dejar un proceso automático de extracción de eventos desde la web.
Por otro lado, con PDF OCR X, la versión de prueba no funciona, por lo que no pudimos llegar a ver un resultado para la herramienta.

ABBY CLOUD	Un año mas! La renovación del tiempo hnce que un año mas se pierda en la vorágine del pasado y pase A confundirse en la corriente de los siglos. El 80 ha cedido su puesto al 81—y, desde ayer, el r.loj del tiempo marca en su inmutable esfera—¡un año mas! Ojalá que la par. esa sabia benéfica que nutre la vida de los pueblos, y de la flue hoy disfrutamos, pueda conservarse - a su sinnnm1 < y -re ' | ' II Sil ■ "HIT I .1 ■ I I-|nO'V>fl feliz esta patria querida de tos Treinta y Tres cuyos hijos «nspirnn A la gran-deza por el trabajo, y 4 la gloria por la libertad.»

Asprise	Un año mas! La renovación del tiempo hace que un año mas se pierda en la vorágine del pasado y pase á confundirse en la corriente de los siglos. El 80 ha cedido su puesto al 81—y, desde ayer, el relej del tiempo marca en su inmutable esfera—¡un año mas! Ojalá que la paz, esa sabia benéfica que nutre la vida de los pueblos, y de la que hoy disfrutamos, pueda conservarse prv.li II .111 ilHUrJlU jui piuMptiu1 j feliz esta patria querida de los Treinta y Tres cuyos hijos «nspiran á la gran deza por el trabajo, y á la gloria por la libertud.»
Dynamsoft OCR SDK	Un año mas! La renovacion del tiempo hace que un año mas se pierda en la voragine del pasado y pase á confundirse en la corriente de los siglos. El SO ha cedido su puesto al 81—y, desde ayer, el reloj del tiempo marca en su inmutable esfera—iun año mas!

	Ojalá que la paz. esa sabia benéfica que nutre la vida de los pueblos, y de la que hoy disfrutamos, eda OINSCTVUTSC feliz esta patria querida de los Treinta y Tres cuyos hijos «aspiran á la gran- deza por el trabajo, y á la gloria por la libertad.»

Tabla 5.4

Finalmente comparamos las herramientas ABBY, Asprise y Dynamsoft (Tabla 5.4). Como vemos en la comparación anterior la herramienta Asprise es un poco mejor que ABBY Cloud ya que palabras como “hace”, “paz”, “los”, “á” las resuelve mejor, mientras que ABBY Cloud para este ejemplo resuelve mejor la palabra “libertad”. Sin embargo la versión de Asprise más barata cuesta USD 5000 mientras que la de ABBY Cloud USD200 por lo que entre estas dos herramientas decidimos quedarnos con ABBY Cloud. Algo similar sucede con Dynamsoft OCR SDK que el resultado obtenido es el mejor de todas las herramientas probadas pero su version mas economica ronda los USD1000, por este motivo decidimos utilizar ABBY Cloud como herramienta para procesamiento OCR en este proyecto.

Finalmente, los OCR a los cuales se les va a hacer una comparación más en profundidad en la siguiente sección son: Tesseract y ABBY Cloud. Tesseract se incluyó en esta prueba por ser una herramienta de código abierto, aunque su resultado fue muy malo como se verá a continuación.

Métrica Accuracy

Generalmente se utilizan dos medidas: la accuracy de las letras y de las palabras. La accuracy de las letras se refiere al porcentaje de aciertos al clasificar la letra correctamente. Análogamente se define para las palabras (Tanner, S., 2004., Deciding whether OCR is feasible).

Para nuestra prueba no utilizamos accuracy por letras, sino que utilizamos por palabras. Pero fuimos un paso más y utilizamos la distancia de Damerau–Levenshtein para la evaluación. El algoritmo que utilizamos compara la distancia palabra a palabra entre el texto original y el resultado de la herramienta. En base a esta distancia, una herramienta es mejor a otra si la mayor cantidad de palabras está cercana a cero. Para nuestro estudio es fundamental medir la correctitud utilizando una medida de distancia y no solamente reportar la accuracy por palabra, ya que saber qué tan incorrectas están las palabras nos da mucha información para luego aplicar una corrección.

Métrica Rouge

Por otro lado, también utilizamos la métrica ROUGE (Recall-Oriented Understudy for Gisting Evaluation). Esta incluye varias medidas para automáticamente determinar la calidad de un resumen al compararlo con su ideal (Chin-Yew Lin, 2004). Estas medidas cuentan el número de superposiciones de palabras ó n-gramas (secuencias de palabras de largo n) entre dos textos. Para nuestro estudio, utilizamos tres medidas: rouge-1, rouge-2[24] y rouge-l[25].

[24] Esta medida contabiliza las superposiciones de n-gramas entre un texto y otro.

[25] Esta medida cuenta el ratio de la cantidad de palabras que hay en la subsecuencia común más larga entre dos textos sobre la cantidad total de palabras. Este resultado permite saber qué tan ordenado están las palabras en el texto.

Evaluación de resultados

Para realizar la evaluación del OCR tomamos un diario de cada editorial y transcribimos a mano un párrafo de cada uno de ellos para luego poder validar contra el resultado del OCR. Esta prueba se realizó sobre 20 tipos de diarios diferentes.

En las siguientes tablas (Tabla 5.5 y Tabla 5.6) se muestra el resultado promedio de las medidas de Rouge, al comparar ABBY Cloud y Tesseract contra el texto original.

ABBY Cloud	Precisión	Recall	Medida F
rouge-1	0.78	0.84	0.81
rouge-2	0.61	0.62	0.62
rouge-l	0.78	0.84	0.81

Tabla 5.5 Resultados métrica Rouge para ABBY Cloud.

Tesseract	Precisión	Recall	Medida F
rouge-1	0.44	0.47	0.45
rouge-2	0.25	0.26	0.26
rouge-l	0.44	0.47	0.45

Tabla 5.6 Resultados métrica Rouge para Tesseract

Como se puede ver, ABBY es considerablemente mejor en las medidas de unigramas y bigramas. Si bien también es mejor en rouge-l, esta medida no aporta mucha información relevante para esta prueba, ya que las herramientas de OCR no intercambian las sentencias (cosa que sí puede

ocurrir en la generación de resúmenes u otras aplicaciones) o las palabras, debido a esto, generalmente rouge-l va a coincidir con rouge-1.

Para la métrica de Accuracy, el resultado de la prueba muestra para cada software el promedio de aciertos utilizando la distancia de palabras antes mencionada desde un largo 0 hasta un largo 5 o mayor. Las imágenes siguientes muestran sobre el eje de las ordenadas la medida Accuracy y sobre el eje de las abscisas la distancia de las palabras.

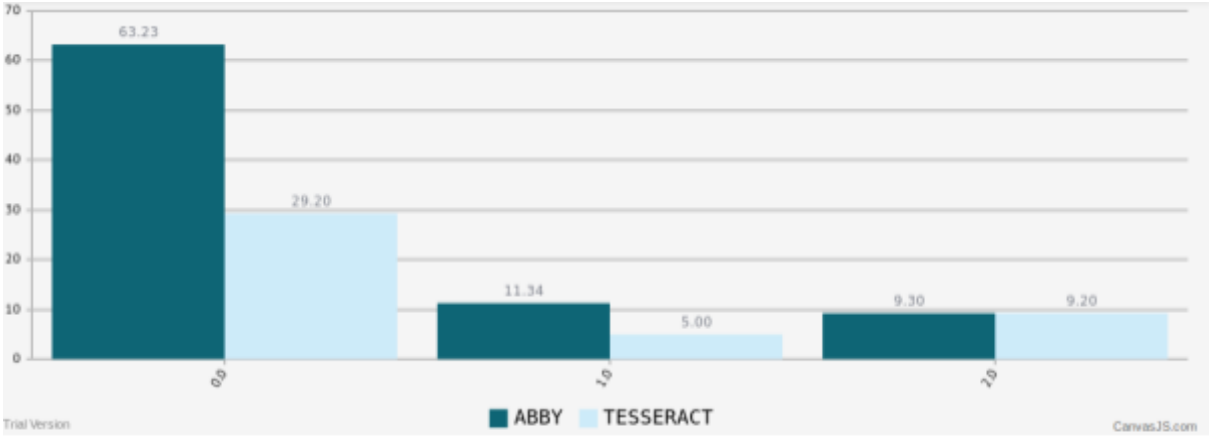


Gráfico 5.1 Métrica Accuracy comparación ABBY-TESSERACT

Como se puede ver en el Gráfico 5.1, ABBY tiene más de un 60% de aciertos en las palabras exactas (largo cero), mientras que Tesseract no alcanza el 30%. También se puede ver que la cantidad de palabras a distancia uno es más del doble en ABBY que en Tesseract, esto es un buen resultado para ABBY ya que significa que corregir los errores a distancia 1 tiene mejores resultados que corregirlos a distancia mayor. Por eso es que buscamos al OCR que más cerca se encuentre de la distancia cero.

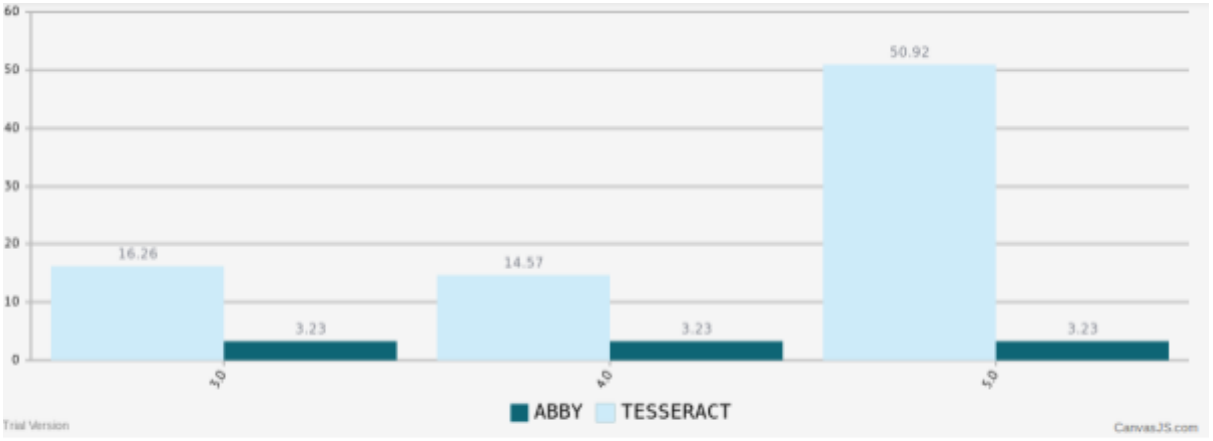


Gráfico 5.2 Métrica Accuracy comparación ABBY-TESSERACT

Por otro lado, en el Gráfico 5.2 tenemos los errores cometidos a distancia mayor que tres. Podemos ver que Tesseract concentra su mayor cantidad de palabras erróneas a un largo mayor o igual que tres. Pero por sobretodo notamos que casi el 51% de la palabras está a distancia mayor o igual a 5 en dicho software, mientras que su competencia apenas supera el 3%.

Por lo que se deduce que ABBY se comporta mucho mejor que su oponente en cualquier de las dos métricas.
A modo de ejemplo se muestra un extracto del diario “EL ECO URUGUAYO” para comparar la salida de estas herramientas:

Texto original:
«Ayer,mis señores,al desaparecer ella, cuando yo volaba detras de los pasos de su maldito caballo, la misteriosa gritó:

Resultado ABBY:
«Ayer, mis.señores, di desaparecer ella, cuando yo volaba (letras de los pasos (le su maldito caballo, la 'misteriosa gritó:

Resultado Tesseract:
«A)'er,m1s sei'10res,:íl desaparecer ella, cuando)o.- _- volaba detras de los pasos de 511 maldito caballo, la misteriosa gritó: . -

5.3. Corrección de los documentos

Debido a la baja calidad de nuestros documentos, el OCR tiende a cometer varios errores. Por otro lado, debido a la mutación del lenguaje de la época, los documentos cuentan con errores ortográficos. Por estos motivos, es necesario corregir la salida del OCR para maximizar la eficiencia de la extracción de información.
El posprocesamiento de los documentos se separa en dos tareas:

- Generación de diccionarios de N-gramas
- Autocorrección de los documentos

5.3.1. Generación de diccionarios de N-gramas

Para poder aplicar el modelo del lenguaje a la autocorrección, es necesario contar con diccionarios de frecuencias de N-gramas. Para generarlos recorremos todos los documentos generando un diccionario de trigramas, uno de bigramas, y uno de unigramas, con los N-gramas válidos respectivamente. Dado un N-grama, lo consideramos válido si todas las palabras que lo componen pertenecen a nuestro diccionario de español. Usamos la herramienta hunspell[26] que posee un diccionario del español y con la cual podemos chequear si una palabra pertenece al mismo.
Estos diccionarios se usarán en la etapa siguiente para corregir los documentos. Es claro ver que mientras más documentos de entrada tengamos, mejor será la calidad de los diccionarios generados y por ende mejor la calidad de la corrección de los documentos.

[26] https://recursospython.com/guias-y-manuales/hunspell-corrector-ortografico/

5.3.2. Autocorrección de los documentos

El proceso para corregir un documento consiste en ir recorriendo las oraciones con una ventana móvil de tres palabras. Si detectamos que alguna de las tres palabras no pertenece a nuestro diccionario del español, le pedimos a hunspell que nos de una lista de palabras candidatas a ser la versión correcta de la misma. A cada una de las palabras candidatas le vamos a dar un puntaje teniendo en cuenta su distancia de Levenshtein con la palabra original y la frecuencia de trigramas, bigramas, y unigramas para los N-gramas que se forman con la palabra candidata y las otras de la ventana. Para saber la frecuencia nos fijamos en los diccionarios de N-gramas que generamos previamente. Luego, si la primer palabra de la ventana necesitaba corregirse, elegimos de entre sus correcciones candidatas aquella que tenga el mayor puntaje. Finalmente avanzamos la ventana una palabra y se repite el proceso. De esta manera, se puede notar que para la corrección de la palabra, influye tanto las dos palabras que se encuentran antes y después de dicha palabra. Cabe destacar que para calcular el puntaje, se le da más peso a los trigramas que a los bigramas, y a los bigramas que a los unigramas.

6. Extracción de eventos

En esta sección se explicará detalladamente la extracción de eventos.

6.1. Solución propuesta

Tomamos como base el artículo ”Unsupervised Techniques for Extracting and Clustering Complex Events in News” (Rusu, D., Hodson, J. and Kimball, A., 2014) donde la forma de extraer un evento se centra en analizar el árbol de dependencias para identificar los verbos y aumentarlos con argumentos en caso de ser necesario. En su solución luego se utiliza OpenIE para restringir los eventos y normalizarlos.

Decidimos crear nuestro propio extractor de eventos, debido a que a priori los extractores para el español que encontramos no tenían la calidad que esperábamos. Más adelante se verá una comparación entre los extractores disponibles para el español y el nuestro.

Nuestra solución está basada en reglas. Incluye el uso de varios recursos y funciones léxicas: POS-Tagging, árboles de dependencia y de parsing, verificación de concordancia (persona, número, género), word embeddings, valencia de verbos.

Los eventos extraídos están compuestos por un verbo y sus argumentos. Cada evento tiene que contener al menos un argumento, es decir que no soportamos verbos avalentes. Si bien puede ser importante encontrar eventos con verbos como ‘llover’ o ‘nevar’, aceptar verbos con valencia nula hace que se genere ruido en la solución y se acepten otros verbos a los cuales no se les pudo encontrar sus argumentos. Debido a ésto y a la poca frecuencia de este tipos de verbos, se decidió no tenerlos en cuenta.

6.1.1. Análisis lingüístico de la oración

Dada una oración necesitamos identificar los verbos y sus argumentos, para ello utilizamos árboles de dependencia (Gramáticas formales para el lenguaje natural,2017). Los árboles de dependencia nos permiten establecer relaciones en una oración. Estas relaciones están dadas entre pares de palabras y de manera asimétrica, es decir una palabra gobierna a la otra. Además cada una de estas relaciones tienen una etiqueta asociada que indica el tipo de relación entre las dos palabras. Entre las etiquetas más conocidas se encuentran sujeto, objeto directo, objeto indirecto, complemento circunstancial, entre otras.

6.1.1.1. Word Embeddings

Word embedding es el nombre de un conjunto de técnicas utilizadas en procesamiento de lenguaje natural mediante las cuales las palabras o frases del vocabulario son mapeadas a vectores de números reales. Estos vectores nos dan una representación semántica de las palabras los cuales nos son útiles para comparar las mismas.

Uno de los algoritmos más populares para realizar word embedding es word2vec. El modelo word2vec es una red neuronal de dos capas que se entrena para reconstruir contextos lingüísticos de palabras. Word2vec recibe un corpus bastante grande como entrada, y produce un espacio vectorial, que suele ser de varios cientos de dimensiones, en donde a cada palabra del corpus le corresponde un vector del espacio. Las palabras que tengan un contexto en común tienen a sus correspondientes vectores cercanos en el espacio.

Los vectores que utilizamos son los que ya vienen pre-entrenados con la herramienta spaCy (que utiliza el algoritmo word2vec).

6.1.1.2. Árbol de dependencia aumentado

La limitación de los árboles de dependencia es que una palabra no puede estar gobernada por más de una, por lo que en el ejemplo de la Figura 6.1 se ve cómo el sujeto de “ovacionó” es “multitud”, pero como “multitud” está

gobernada por “aplaudió” esta queda sin sujeto. Un caso similar ocurre con el objeto directo. Por la anterior limitación en algunos casos es necesario aumentar las dependencias.

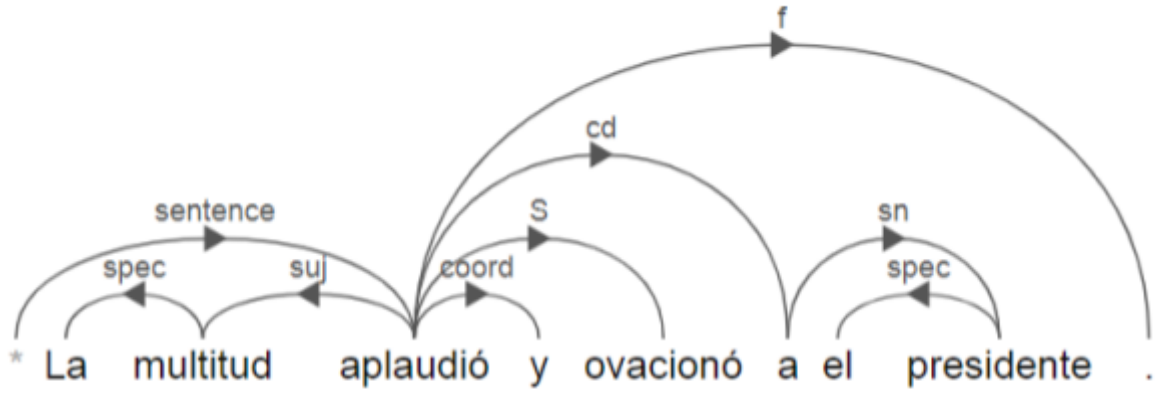


Figura 6.1

De aquí en adelante cuando mencionemos verbo-ancestro y verbo-descendiente haremos referencia a dos verbos conectados mediante un camino en el árbol de dependencia que va desde el verbo-ancestro al verbo-descendiente. En el ejemplo de la Figura 6.1 “aplaudió” es el verbo-ancestro de “ovacionó” (verbo-descendiente). Para este proyecto decidimos aumentar las siguientes etiquetas: sujeto, objeto directo y complementos circunstanciales. Para los casos en los cuales el verbo-descendiente carezca de sujeto decidimos agregarle el sujeto del verbo-ancestro cuando hay concordancia entre su sujeto y el verbo-descendiente en número, persona y género. Por ejemplo, en el caso de la Figura 6.1 se aumenta el verbo-descendiente “ovacionó” con el sujeto del verbo-ancestro “multitud” .

La concordancia gramatical es una relación entre dos unidades gramaticales y se manifiesta en diferentes niveles: como género, número y persona. Existen dos tipos de concordancia, nominal y verbal. La concordancia nominal, se define como la coincidencia del género y del número por ejemplos en los casos:

- Un sustantivo con el determinante o los adjetivos que lo acompañan

- Un pronombre y su antecedente o consecuente

La concordancia verbal, se define como la coincidencia de número y persona entre un verbo y su sujeto. En nuestro caso para determinar la concordancia del verbo-descendiente con el sujeto del verbo-ancestro obtenemos los tres rasgos sintácticos del sujeto(persona, género y número) a partir de sí mismo, su determinante y el verbo-ancestro y los comparamos con los rasgos del verbo-descendiente.

Por otro lado para aumentar el objeto directo del verbo-descendiente al verbo-ancestro utilizamos diferentes técnicas. La primer técnica utilizada es la del estudio de la valencia de los verbos, para esto utilizamos AnCora (Ancora|Corpus) que es un corpus del español anotado que cuenta con esta información. Alguna de esta información es: el lema y categoría morfológica, constituyentes y funciones sintácticas, estructura argumental, papeles temáticos y clase semántica verbal, entre otros. Dispone de un léxico verbal para el español de 2.647 entradas que contiene para cada entrada lo siguiente:

- la clase semántica
- subcategorización
- estructura argumental y papeles temáticos.

Es en este léxico que nos vamos a basar para determinar la valencia de cada verbo, y más específicamente en la estructura argumental. Para la estructura argumental AnCora anota al verbo en el tag lss (por sus siglas en inglés, Lexical Semantic Structures) con la clasificación del verbo: transitivo, ditransitivo, etc (AnCora|Estructura argumentos). Esta clasificación determina la cantidad de argumentos del predicado verbal y el tipo de sus argumentos. Para cada clasificación AnCora define reglas de cómo deben ser sintácticamente el sujeto y los demás complementos del verbo.
Por ejemplo: El verbo “decantó” tiene como regla asociada que el complemento directo sea siempre una preposición.

En esta etapa utilizamos esta información para aumentar los verbos, si el verbo-ancestro no tiene un objeto directo asociado se busca el verbo-ancestro en el corpus de AnCora para determinar su valencia. Este verbo puede tener más de una diátesis, es decir el mismo verbo puede tener diferente cantidad de argumentos, por lo que se estudia caso a caso.

Caso uno, si hay unanimidad en la necesidad de objeto directo se agrega el objeto directo del verbo-descendiente al verbo-ancestro.

Caso dos, si FreeLing no encontró un objeto directo, tomamos la diátesis más frecuente en AnCora y verificamos que cumpla con su regla asociada. Si la cumple, entonces se le agrega el objeto del verbo-descendiente al verbo-ancestro.

Caso tres, si el verbo no es encontrado en el corpus o si existe al menos una regla asociada a ese verbo que necesite del objeto directo se resuelve utilizando word embedding con cierto umbral de similitud entre el objeto directo del verbo-descendiente con el verbo-ancestro. Un ejemplo de este caso se muestra en la Figura 6.2 donde al analizar con word embedding las palabras “administra” (verbo-ancestro) y “empresas” (objeto directo del verbo-descendiente) se determina que al evento que tiene como verbo “administra” se le agrega el objeto directo “empresas”.

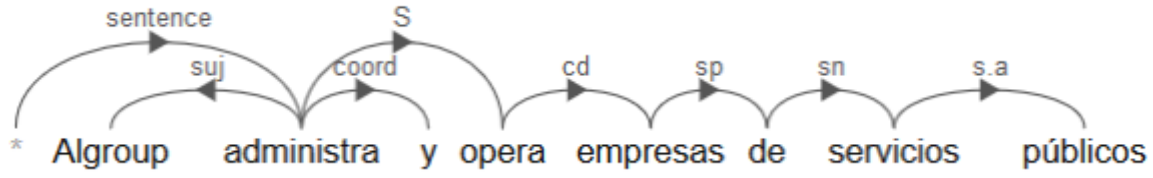


Figura 6.2

Entonces, dependiendo de qué tan similares sean semánticamente es que decidimos si agregarlo o no.

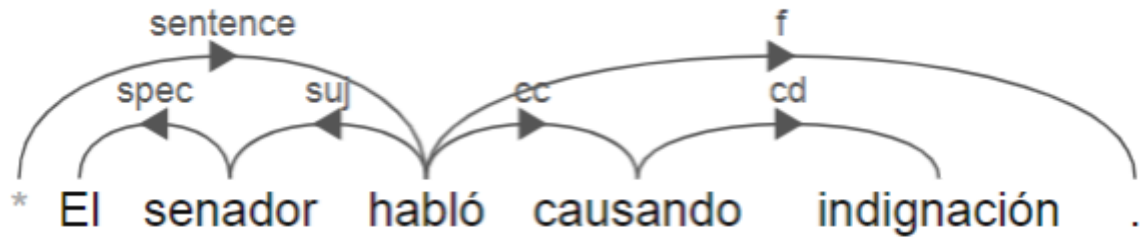


Figura 6.3

Para los casos en los que el verbo-descendiente sea un gerundio o un participio, no aumentamos el objeto directo del verbo-ancestro, por ejemplo si no existiera esta regla, para el caso de la Figura 6.3 se generarían los eventos $< El\ senador,\ causando,\ indignación >$ y $< El\ senador,\ habló,\ indignación >$ cuando este último debería ser $< El\ senador,\ habló >$.

Tampoco lo aumentamos cuando no cumple ninguna regla de AnCora.

Por otro lado aumentamos los complementos circunstanciales que sean entidades del tipo locación y argumentos temporales, tanto del verbo-ancestro al verbo-descendiente como del verbo-descendiente al verbo-ancestro siempre y cuando no tengan ya uno propio.

6.1.1.3. Resolución de correferencias

La correferencia es la referencia al mismo tiempo de dos o más expresiones lingüísticas en el mismo texto. Por ejemplo en la oración:

“Juan come la pizza que a Pedro le gusta.”

Donde “que” es un pronombre relativo y hace referencia a “pizza” y “le” es un pronombre personal y hace referencia a Pedro.

El problema de las correferencias es un problema aún no resuelto. Probamos con la herramienta FreeLing y no dio buenos resultados, por lo que decidimos implementar una solución sencilla, pero no completa. No es completa, debido a que está basada únicamente en la resolución de correferencias de pronombres relativos. Para resolverla, utilizamos el árbol de parsing para encontrar todos los sintagmas nominales de la oración. Luego sustituimos el

pronombre relativo, por el sintagma nominal más cercano a él desde la izquierda, que cumpla con la concordancia de los tres rasgos sintácticos, entre el verbo del pronombre y el sustantivo raíz del sintagma nominal (aumentando los rasgos con su determinante y su verbo).
La resolución de correferencia la utilizamos cuando el pronombre relativo es un argumento de un evento (ya sea como sujeto o como objeto).

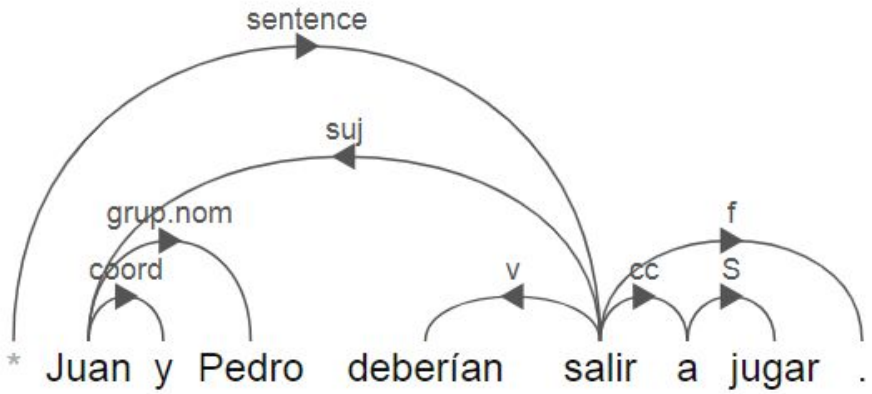
6.2. Evaluación de herramientas de parsing

Para realizar este proyecto el requisito fundamental que debía cumplir la herramienta de parsing es que tuviera soporte para el español. Las herramientas encontradas que cumplan este requisito fueron: Freeling[27], MaltParser[28], spaCy[29] y Stanford Parser[30].
Stanford Parser tiene soporte para español, pero para este idioma no genera el árbol de dependencia que es imprescindible para nuestro trabajo. Por este motivo queda descartado.
Para las restantes tres herramientas probamos un conjunto de sentencias y analizamos los resultados para decidir con cual trabajar. Algunos de los resultados interesantes a analizar son los siguientes:

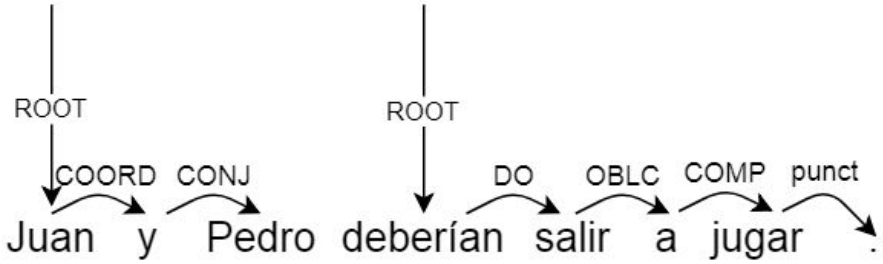
[27] http://nlp.lsi.upc.edu/freeling/demo/demo.php
[28] http://www.maltparser.org/
[29] https://spacy.io/
[30] https://nlp.stanford.edu/software/lex-parser.shtml

“Juan y Pedro deberían salir a jugar.”:

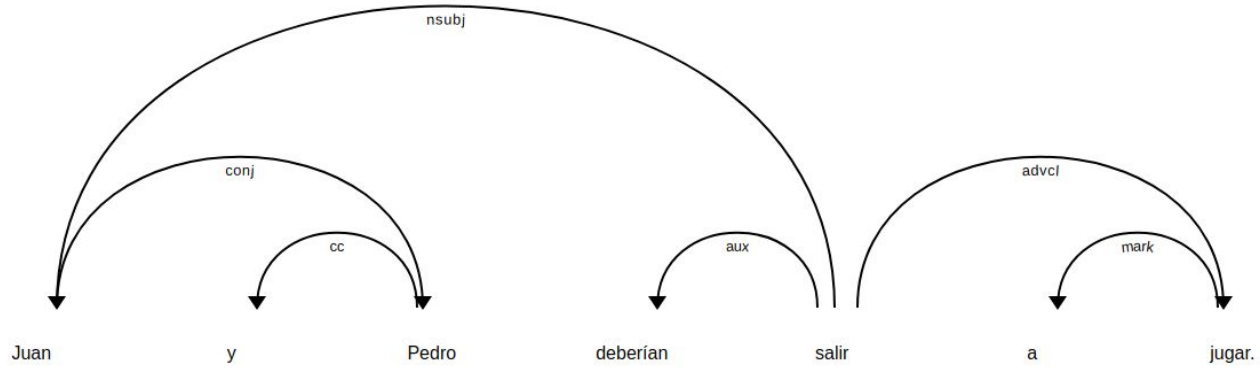
Freeling:



MaltParser:



spaCy:

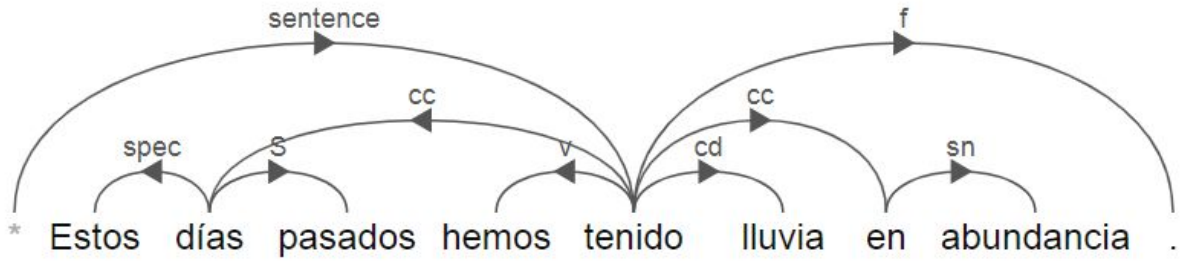


Para este caso Freeling y spaCy marcan como root de la oración a “salir” y resuelven bien el sujeto, mientras que MaltParser marca como root de la

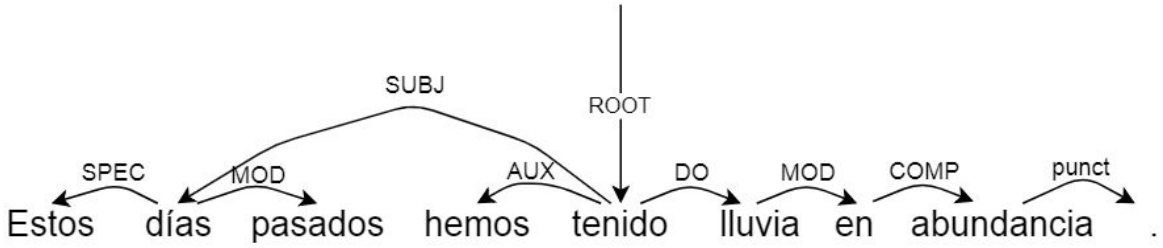
sentencia a “deberían” y deja a “Juan y Pedro” en un subárbol independiente lo cual es incorrecto.
Con respecto a “a jugar” Freeling lo marca correctamente como complemento circunstancial de “salir”, mientras que spaCy le asigna un advcl (cláusula adverbial).

“Estos días pasados hemos tenido lluvia en abundancia.” :

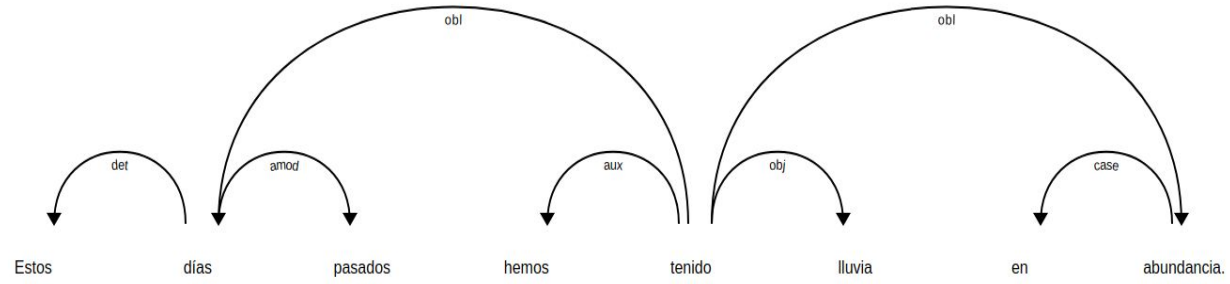
Freeling:



MaltParser:



spaCy:

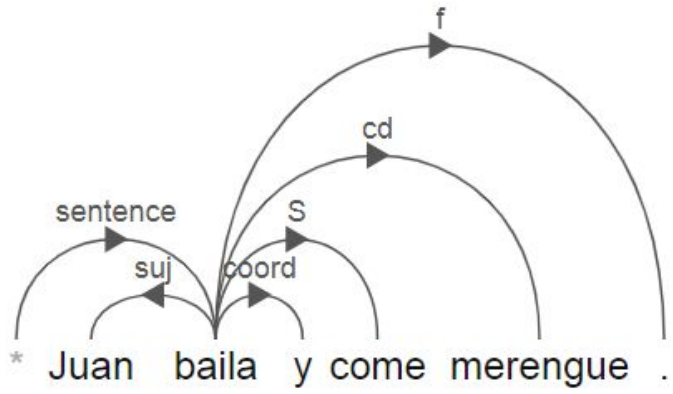


En este caso Freeling y spaCy lo hacen de forma correcta y muy similar, mientras que MaltParser marca incorrectamente como sujeto a “Estos días

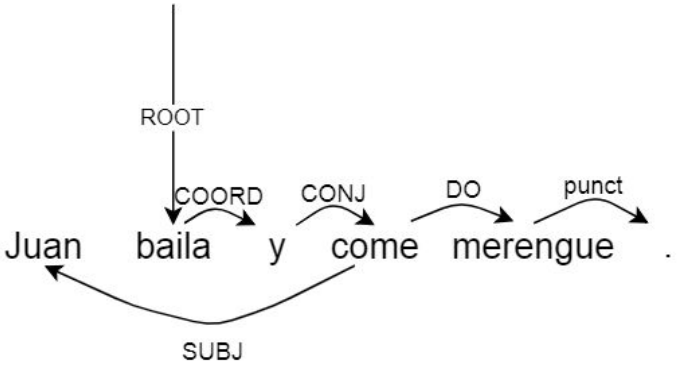
pasados” y “en abundancia” como modificador de lluvia cuando debería ser un complemento circunstancial del verbo “tenido”.

“Juan baila y come merengue.”:

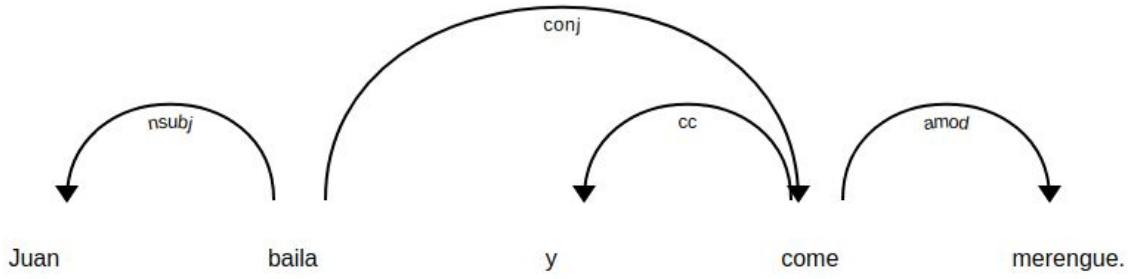
Freeling:



MaltParser:



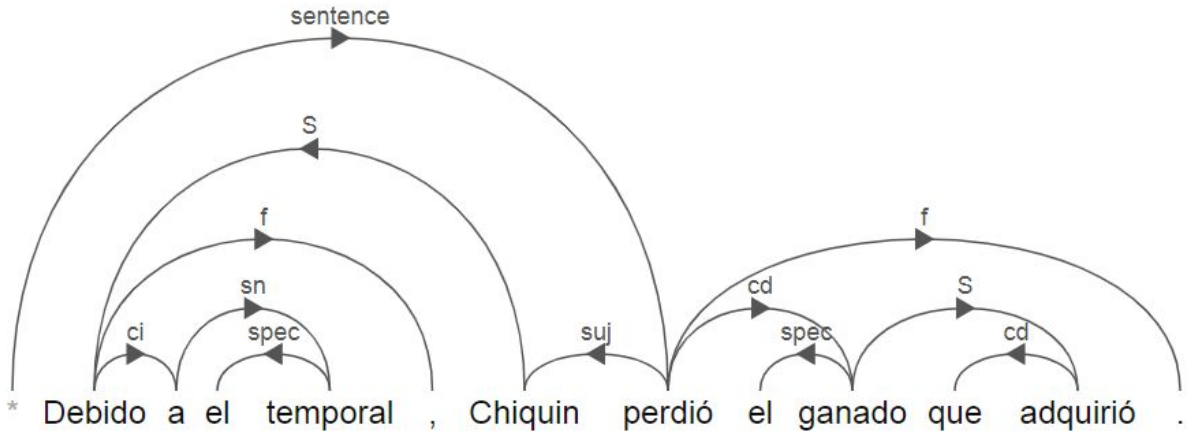
spaCy:



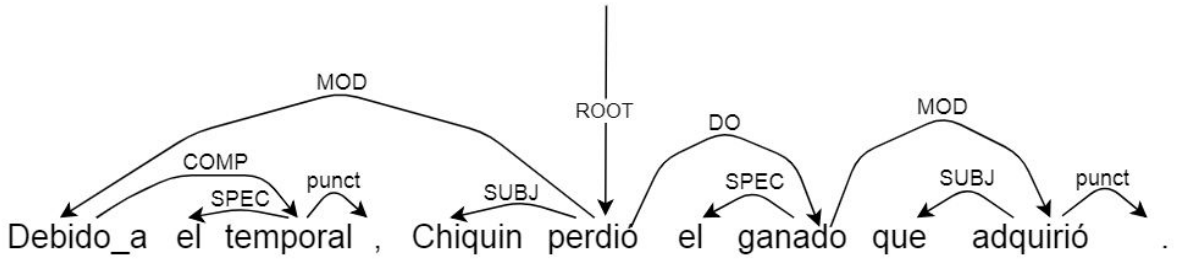
En este caso por la estructura de la sentencia, semánticamente es ambigua incluso para un humano. Juan puede estar bailando merengue y comiendo merengue, o bailando cualquier tipo de música y comiendo merengue. Por lo que el resultado correcto no es único.
De todos modos, en el caso de Freeling asigna “merengue” como objeto directo de “baila” y deja “come” como una sentencia independiente. Cabe destacar que cuando se suplanta “merengue” por otra comida que no sea un tipo de música, Freeling asigna como objeto directo de “come” a la nueva comida ingresada, haciendo parecer que Freeling hace un control semántico sobre las dependencias que va a asignar a los verbos.
Para este caso el resultado realizado por MaltParser no es correcto ya que “baila” no tiene de sujeto a “Juan”. Con respecto a spaCy resuelve correctamente a “Juan” como sujeto de “baila” pero marca a “merengue” como modificador de “come” cuando debería ser un complemento directo.

“Debido al temporal, Chiquin perdió el ganado que adquirió.”:

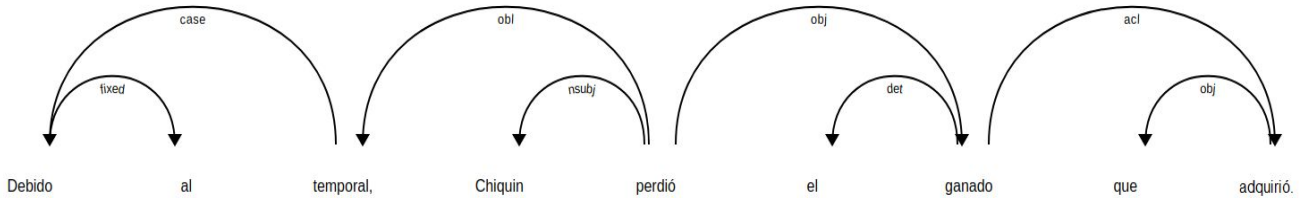
Freeling:



MaltParser:



spaCy:



En este ejemplo la dificultad radica en la oración subordinada “que consiguió”, que Freeling y spaCy resuelven bien asignando a “que” como objeto directo, ya que hace referencia a “el ganado”. Mientras que MaltParser asigna erróneamente a “que” como sujeto.

Después de analizar varias oraciones con las tres herramientas de parsing decidimos descartar a MaltParser ya que algunas formas de resolver las dependencias no son correctas. Mientras que Freeling y spaCy retornan un resultado similar para la mayoría de las oraciones, por lo que el criterio final en la elección de la herramienta de parsing se basó en el conocimiento sobre la misma. Al ya haber utilizado Freeling en ocasiones anteriores, terminamos optando por esta.

6.3. Evaluación de herramientas OpenIE

En la sección 3.2.1 entre otras cosas se habló sobre OpenIE y las herramientas que hay para el español. Aquí haremos una evaluación de esas dos herramientas: ArgOE y ExtrHech. Dicha evaluación no pretende ser

exhaustiva. Se analizará cómo se comporta cada una de ellas sobre un conjunto pequeño de oraciones.

Oración 1: *"El mecánico reparó el coche."*
Este ejemplo de oración es el más básico. Está bien marcado el sujeto, verbo y objeto de la relación, o semánticamente hablando el agente y paciente.

Salida ArgOE: (El mecánico, reparó, el coche)
Salida ExtrHech: No fue encontrada la relación.

Sorprende ver que ExtrHech no haya podido encontrar la relación en una oración que es muy corta, expresada en voz activa y el agente/paciente bien definidos. Por su parte, ArgOE realizó la extracción correctamente.

Oración 2: *"El coche fue reparado por el mecánico."*
La oración dos es semánticamente igual a la primera, solo que ésta se encuentra en voz pasiva.

Salida ArgOE: (El coche, fue reparado por, el mecánico)
Salida ExtrHech: No fue encontrada la relación.

Aquí de nuevo, ExtrHech no pudo extraer la relación. Por su parte, ArgOE extrajo la relación. Cabe destacar que el primer argumento '*el coche*' es el sujeto a nivel sintáctico, pero a nivel semántico es el paciente. Se hace esta distinción porque luego para clusterizar es importante conocer los roles semánticos y no los sintácticos, por lo que si bien ArgOE encuentra la relación, no lo hace de la forma deseada para nosotros.

Oración 3: *"Juan come pizza, mientras que Pablo baila salsa."*

La tercera oración es un tipo de oración coordinada por la palabra *'y'*. Contiene dos verbos: *come* y *baila*. En nuestro proyecto nos interesa que dos eventos sean reconocidos: *"Juan come pizza"* y *"Pablo baila salsa"*.

Salida ArgOE: (Juan, come, pizza), (Pablo, baila, salsa)
Salida ExtrHech: (Juan, come, pizza), (Pablo, baila, salsa)

Para ésta prueba, ambos extractores hallaron correctamente las dos relaciones que habían.

Oración 4: *"Roberto asaltó el banco y se llevó todo el dinero."*
La cuarta oración es un tipo de oración subordinada. Existe una relación de dependencia entre la primer oración y la segunda unidas por la conjunción 'y'. Esta oración contiene dos relaciones o eventos.

Salida ArgOE: (Roberto, asaltó, el banco), (Roberto, se llevó, todo el dinero)
Salida ExtrHech: (Roberto, asaltó, el banco)

En este caso, ArgOE logra encontrar ambas relaciones, colocándole el sujeto a la segunda. Mientras que ExtrHech solamente encuentra la primera.

Oración 5: *"El Presidente de la República asumió el cargo que dejó Ana."*
La última oración contiene un sujeto compuesto con una preposición, pero lo más importante es que tiene un pronombre relativo '*que*' el cual hace referencia al cargo presidencial.

Salida ArgOE: (El Presidente de la República, asumió, el cargo), (el cargo, dejó, Ana).
Salida ExtrHech: (El Presidente de la República, asumió, el cargo)

Finalmente, para esta última oración, ExtrHech no logra resolver la coreferencia y por ende solo encuentra el primer evento. ArgOE exitosamente encuentra ambos eventos, sustituyendo para el segundo la coreferencia '*que*' por '*el cargo*'.

Como conclusión de la evaluación se aprecia que para cada oración de este conjunto pequeño y acotado, ArgOE se comporta igual o mejor que su competencia. Tal vez la diferencia de performance radica en la forma de trabajo de cada una de estas herramientas. ArgoOE utiliza la información de los árboles de dependencia (dependency parsing), mientras que ExtrHech utiliza el POS Tag (shallow parsing).
Sin embargo en el artículo "Multilingual Open Information Extraction" (Gamallo, P. and Garcia, M., 2015) se evalúan ambas herramientas obteniendo los resultados que aparecen en la Figura 6.4 donde se muestra un mejor resultado para ExtrHech con respecto a ArgOE utilizando un dataset extraído de la web.

Systems	correct extractions	total extractions	Precision (%)
argoe	107	214	50%
extrahech	-	-	55%

Figura 6.4

Por otro lado, en el artículo "Open Information Extraction for Spanish Language based on Syntactic Constraints" (Zhila, A. and Gelbukh, A., 2014), ExtrHech utiliza un corpus de noticias[31] para evaluar su extractor, por lo que decidimos aprovechar la similitud de nuestros datos con ese dataset y realizar la comparación contra el mismo conjunto de datos que utilizaron. En la sección 8 (Evaluación de los resultados), se verá esta comparación.

[31] https://www.gelbukh.com/resources/spanish-open-fact-extraction/ver1/News300dataset.zip

7. Clustering

Clustering es el proceso de agrupar entidades similares. El objetivo de esta técnica de aprendizaje automático es encontrar similitudes en los puntos para luego agruparlos.

Agrupar objetos similares ayuda a describir los atributos de distintos grupos. En otras palabras, esto nos da una visión sobre los patrones que hay en distintos grupos de objetos. Hay muchas aplicaciones de agrupación de datos sin etiquetar, por ejemplo, se pueden identificar distintos grupos/segmentos de clientes y aplicar técnicas de mercadeo específicas para cada grupo con el objetivo de maximizar las ventas. Otro ejemplo es el de agrupar documentos que pertenecen a tópicos similares. Clustering también se usa para reducir la dimensionalidad de los datos cuando se está trabajando con un alto número de variables.

Hay muchos algoritmos implementados para ésta técnica. A continuación enumeraremos algunas de las familias de algoritmos, junto con algún ejemplo de cada una, pero explicaremos más adelante en detalle los que en principio consideramos utilizar.

Métodos de Clustering:

Métodos de partición: K-Means, PAM, CLARA, etc.
Clustering Jerárquico: Agglomerative, Divisive, etc.
Métodos de clustering Híbridos: Hierarchical K-Means, HCPC, etc.
Clustering Fuzzy: Fuzzy c-means, etc.
Clustering basado en un Modelo
Clustering basado en densidad: DBSCAN, etc.

Validación y Evaluación del Clustering

Las estrategias de validación y evaluación del clustering, consisten en medir qué tan buenos son los resultados del clustering. Antes de aplicar un algoritmo de clustering a un conjunto de datos, lo primero que se debe hacer es determinar la tendencia de los datos a ser clusterizables. Es decir, estudiar si los datos contienen una estructura de agrupación inherente. Para esto se pueden utilizar métodos estadísticos (por ejemplo la estadística Hopkins), o métodos visuales (por ejemplo el algoritmo VAT).

En caso de que los datos tengan tendencia a ser clusterizables, se puede determinar la cantidad óptima de clusters. Esto es útil si se quiere usar un algoritmo de clusterización que requiera pasarle de antemano el número de clusters como los métodos de partición. Los métodos para determinar la cantidad óptima de clusters se dividen en dos grupos: métodos directos (Elbow Method, Silhouette Method) y métodos estadísticos (Gap Statistic Method).

Finalmente, se pueden usar una variedad de medidas para evaluar qué tan buenos son los resultados del clustering. Por ejemplo para nuestra evaluación utilizamos las siguientes métricas:

- **<u>Silhouette Coefficient:</u>**
    - **Fórmula**:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

    - **Definición**: $a(i)$ es la media de la distancia entre el punto i y el resto de puntos de su cluster mientras que $b(i)$ es la media de la distancia entre el punto i y los puntos del cluster más cercano. Para determinar el coeficiente del sistema, hay que calcular la media de este valor para cada punto.
      El coeficiente va entre -1 y 1. Si está cercano a 1, indica que el

sistema está bien clusterizado, mientras que si está cercano a -1, significa que el sistema está muy mal clusterizado. Valores cercanos a 0 significan que hay algo de solapamiento entre los clusters.

- **<u>Pureza:</u>**
  - **Fórmula:**

$$\frac{1}{N}\sum_{m\in M}\max_{d\in D}|m\cap d|$$

  - **Definición:** m son la cantidad de puntos en el cluster M, d son los puntos del tipo D.

    La pureza de un cluster se mide contando la máxima cantidad de puntos de un mismo tipo que hay en un cluster sobre la cantidad total de puntos de ese cluster. Para calcular la pureza del sistema se calcula la pureza de cada cluster y se divide entre la cantidad total de puntos. El valor va entre 0 y 1 y representa el porcentaje de pureza del sistema.
- **<u>Entropía:</u>**
  - **Fórmula:**

$$H=\sum_{i\in C}H(i)\frac{N_i}{N}\quad H(i)=-\sum_{j\in K}p(i_j)\log_2 p(i_j)$$

  - **Definición:** H representa la entropía del sistema, $H(i)$ es la entropía del cluster i, N es la cantidad total de datos en el sistema, N_i es la cantidad de datos en el Cluster i, $p(i_j)$ es la probabilidad de la clase j en el cluster i (es simplemente el porcentaje de puntos de esa clase en ese cluster).

    La entropía se usa en los algoritmos de clustering como medida de la diversidad o heterogeneidad de un conjunto de objetos.

A continuación mencionaremos los dos algoritmos que consideramos utilizar:

1. K-mean Clustering
2. Chinese Whispers Clustering

K-mean Clustering

1. Tiene como entrada al algoritmo un número K, que representa la cantidad de clusters que se desea encontrar. Sitúa K centroides de manera aleatoria en lugares del espacio.
2. Utilizando la función de distancia entre dos puntos del espacio y los centroides, se asigna cada punto al cluster más cercano.
3. Recalcula el centro de cada cluster como un promedio de los puntos en el espacio asignados a él.
4. Repite pasos 2 y 3 hasta que no haya más cambios.

Como se mencionó anteriormente, hay formas de determinar la cantidad óptima de clusters. Sin embargo, nosotros vamos a utilizar un K = secciones de los diarios uruguayos actuales. Por ejemplo: policiales, economía, sociedad, política, judiciales, deportes, entre otros.

Chinese Whispers

El algoritmo de Clustering: Chinese Whispers (Biemann, C.2006). La ventaja del algoritmo de Clustering de grafos es que es escalable, y que a diferencia de otros algoritmos de Clustering, no hay que predefinir la cantidad de clusters en que se van a agrupar los datos.

El algoritmo es muy simple y se detalla a continuación.

1. Se le asigna a cada evento una clase propia.
2. Se randomiza el orden de los eventos.
3. A cada evento se le asigna la clase del evento más similar a él.

4. Se repiten los pasos 2 y 3 hasta que el algoritmo converja o por un predeterminado número de iteraciones.

7.1. Clustering de eventos

Similitud entre Eventos

Para poder realizar el clusterizado de los eventos, necesitamos definir una medida de distancia o similitud de eventos. Nuestros eventos están formados por un verbo y sus argumentos, por lo que nuestra medida de similitud entre dos eventos va a ser una combinación de las similitudes entre las palabras que conforman cada evento.

Para la similitud entre palabras vamos a investigar dos medidas distintas, una es utilizar word embeddings, y la otra es usar una similitud de camino haciendo uso de WordNet. Una similitud de camino computa el menor número de vértices entre dos palabras, asumiendo una estructura jerárquica como la de WordNet (básicamente un grafo). En general, dos palabras con un camino de vértices corto son más similares entre sí que dos con un camino de vértices largo.

WordNet trae varias medidas de similitud de este tipo definidas, en particular la que tomamos fue la similitud de Leacock-Chodorow Similarity (Leacock, C., & Chodorow, M, 1998) que tiene en cuenta la distancia entre las palabras en la estructura de WordNet y la altura de ambas en la estructura jerárquica.

Realizamos pruebas utilizando ésta distancia, pero trajo ciertas limitaciones. Por ejemplo, no permite comparar un verbo con un sustantivo. Un verbo puede contener información semántica y si no permite realizar dicha comparación, se pierde información valiosa. Por otro lado, tampoco estaba muy claro cómo calcular la distancia entre frases.

Veamos ahora entonces el método para comparar dos eventos.

Según en lo que ponga el foco la frase, el sujeto y el objeto podrían estar intercambiados en dos frases que representan eventos idénticos. Asimismo, hay veces que un concepto puede aparecer como verbo o como sustantivo (por ejemplo llovió y lluvia).
Es por esto que no podemos simplemente comparar sujeto con sujeto, verbo con verbo y objeto con objeto, para determinar la similitud entre dos eventos.

Lo que hacemos entonces es generar todas las combinaciones posibles entre el verbo y sus argumentos de un evento y comparala con las combinaciones del otro evento(6 combinaciones en el caso de que cada evento esté formado por 1 verbo y 2 argumentos).
<v1, suj1, obj1> <v2, suj2, obj2>
<v1, suj1, obj1> <v2, obj2, suj2,>
<v1, suj1, obj1> <suj2, v2, obj2>
<v1, suj1, obj1> <suj2, obj2, v2>
<v1, suj1, obj1> <obj2, v2, suj2>
<v1, suj1, obj1> <obj2, suj2, v2>
La similitud de cada una de estas combinaciones se calcula sumando la similitud de comparar elemento a elemento posicional.
Para el cálculo de la similitud de dos elementos, como están compuestos por múltiples palabras, lo que se hace es calcular el vector centroide (o media) de esas palabras quitando las stop words[32]. Para calcular el centroide, lo que se hace es sumar los vectores de las palabras que lo componen coordenada a coordenada, para luego dividirlo entre la cantidad total de palabras. Una vez hallados los vectores centroides de cada argumento a comparar, se procede a calcular la similitud entre ellos, utilizando la función de similitud coseno.
Una vez calculada la similitud de estas combinaciones, nos quedamos con la de mayor similitud.

[32] Las stop words son palabras muy comunes en el lenguaje y por ende no aportan información muy relevante

7.2. Etiquetado de clusters

Una vez construidos los clusters, nuestro objetivo es etiquetarlos. De aquí surgieron dos posibles alternativas, según lo que deseara extraer el usuario.
Una alternativa podría ser que el usuario proporcione una lista de términos de su interés y nosotros devolverle los clusters rankeados en base a la similitud con la lista de términos.
La otra alternativa, la cual terminamos eligiendo, fue la de extrapolar las secciones o categorías de los diarios, tal cual las conocemos hoy, a los diarios antiguos. Al no contar los diarios antiguos con secciones y dado que hoy en día es natural ver las noticias agrupadas por las mismas, nos pareció una buena idea intentar realizar esta agrupación.
Lo primero que hicimos, fue scrapear los diarios digitales: El País, LaRed21 y Montevideo Portal. Como las categorías no son exactamente iguales, hubo que hacer un trabajo manual para generalizarlas y agruparlas a través de los diarios. Las categorías elegidas en definitiva fueron: Política, Economía, Sociedad, Policiales y Clima. Para el caso de la categoría Clima además contábamos con un glosario[33] de términos provisto por la FIC (Facultad de Información y Comunicación) el cual se agregó al corpus de noticias del clima. Este glosario consiste en el término o concepto a definir, sinónimos o significados diferentes para diferentes países y descripción del término. Para nuestro caso, este glosario es importante ya que no hay una sección de noticias dedicadas a eventos climáticos.
Luego de generalizar las categorías, lematizamos todas las palabras de cada sección para que cuando se realice la comparación se haga lema a lema. Esto es útil ya que si estoy buscando si la palabra “lloviendo” aparece en una categoría y aparece la palabra “llovió” se comparan ambos lemas (“llover” en ambos casos) y resulta en un acierto. También se quitaron las stop words.

[33] https://drive.google.com/file/d/0B19WqTgkg6aHekk0OW1LZ0N4NkU/view?usp=sharing

Una vez terminada la fase de scraping y lematizado, lo que necesitamos es comparar cada categoría o sección contra cada cluster para etiquetarlo. Existen varias formas de hacer esto:

- Uno forma es la de contar la frecuencia de las palabras de los clusters, para cada categoría (quitando las stop words), y asignar la etiqueta de la categoría que cuente con la mayor frecuencia.
- Otra forma es seleccionar de manera manual los X términos más relevantes de cada sección y contar las ocurrencias de éstos términos en cada cluster.
- Una tercera opción es generar un vector que represente cada sección utilizando sus documentos. Luego se comparan los vectores de las secciones con los vectores centroides de cada cluster y a quien tenga menor distancia, se le asigna esa etiqueta.

Dentro de estas tres opciones seleccionamos la primera por ser la más sencilla y no ser el módulo principal del sistema para nosotros. Investigar otras opciones de etiquetado quedará para trabajos futuros.

8. Evaluación de los resultados

A continuación detallamos los resultados de la evaluación de los módulos de extracción de eventos, clustering y etiquetado. La herramienta de OCR utilizada (Abby OCR) se evaluó en profundidad en la sección 5.2.4. Evaluación de herramientas.

Para la evaluación del módulo de extracción de eventos utilizamos la técnica descrita en el artículo "Open Information Extraction for Spanish Language based on Syntactic Constraints" (Zhila, A. and Gelbukh, A., 2014), en donde se parte de un corpus[34] de noticias separado en trescientas oraciones. Estas trescientas oraciones forman parte de un corpus más grande llamado: "News Commentary Parallel Corpus"[35], que se utiliza para evaluar tareas de traducción automática. Los autores del artículo lo que hicieron fue ejecutar su extractor sobre estas trescientas oraciones y luego dos anotadores humanos etiquetaron si era correcta o no la extracción de eventos. Por lo tanto, este corpus de trescientas oraciones cuenta también con los eventos extraídos por ellos. Nosotros utilizamos el mismo método, es decir ejecutamos nuestro extractor sobre las trescientas oraciones dando como resultado la extracción de trescientos ochenta y seis eventos, luego evaluamos si los eventos extraídos eran correctos o incorrectos para cada oración.
Cabe destacar, que dado como realizaron la evaluación los autores, no existía un gold standard creado por anotadores humanos, sino que anotaban sobre el resultado del extractor, por lo que la falta de éste hizo que imitáramos la forma de evaluación.
Finalmente, calculamos las medidas de Precisión y Recall las cuales se muestran en la Figura 8.1

[34] https://www.gelbukh.com/resources/spanish-open-fact-extraction/ver1/News300dataset.zip
[35] http://www.casmacat.eu/corpus/news-commentary.html

Precision	Recall
56%	70%

Figura 8.1

Para el mismo corpus, la herramienta ExtrHech obtuvo los resultados que se muestran en la Figura 8.2.

Dataset	Precision	Recall
News Commentary	0.59	0.48

Figura 8.2

Como se puede ver, ExtrHech extrae los eventos con una Precisión del 59% y Recall del 48%, mientras que el módulo de extracción implementado en este proyecto lo hace con un 56% y 70% respectivamente. Cabe destacar que debido a que la anotación del corpus no era la misma, no se puede afirmar fehacientemente que la comparación sea correcta. También hay que destacar que consideramos correcto los eventos extraídos si y solo si tenían todos sus argumentos, considerando de esta manera incorrectos los eventos parciales.
Nuestra solución tiene un alto Recall, debido a que cualquier verbo que se encuentre en una oración, para nosotros es un posible evento, siempre y cuando no exista otro que lo contenga. ExtrHech, solamente encuentra eventos, si existen ambos argumentos del mismo, mientras que nuestra solución con al menos encontrar uno de los argumentos, ya retorna un evento.

Para la evaluación del clustering extrajimos los eventos de cien oraciones de diferentes dominios (política, policiales, sociedad, clima y economía) de noticias de la web. A cada oración le asignamos la etiqueta correspondiente al dominio en donde se encuentra. Esto se hace como una forma automática de anotar el corpus. Una vez anotadas las oraciones y extraídos los eventos de las mismas, se clusterizan éstos. A partir de los clusters generados y la etiqueta

asignada evaluamos los clusters, utilizando las siguientes medidas (Eréndira Rendón et al, 2011):

1. Silhouette Coefficient
2. Pureza
3. Entropía

En la Figura 8.3 se muestran los resultados.

SILOUHETTE COEFFICENT	PURITY COEFFICENT	ENTROPY COEFFICENT	LABELING COEFFICENT
0,01	0,57	1,41	0,29

Figura 8.3

El valor 0,01 del coeficiente de Silhouette nos indica que existe solapamiento entre los clusters. La pureza y la entropía dieron un resultado aceptable. La primera nos indica que en promedio, más de la mitad de los elementos de un cluster pertenecen al mismo tipo. Por otro lado, la entropía nos indica que los clusters tienen un grado aceptable de homogeneidad

Para evaluar el etiquetado, vamos a calcular una medida que le llamamos Labeling Coefficient. El Labeling Coefficient para un cluster se calcula como:

$$LC = \frac{\sum_c f(c)}{TotalClusters} \qquad f(c) = \begin{cases} 1 & \text{si la estiqueta del cluster es correcta} \\ 0 & \text{si no es correcta} \end{cases}$$

El etiquetado de un cluster es correcto, si la etiqueta asignada coincide con la etiqueta predominante entre los eventos que lo componen. En la Figura 8.3 se muestra el resultado logrado.
Es coherente el valor del coeficiente de etiquetado con el grado de solapamiento reportado por el coeficiente de Silhouette.

9. Conclusiones y Trabajo futuro

9.1. Conclusiones y problemas encontrados

Se logró construir un sistema de extracción de eventos a partir de textos de prensa en español, el cual presenta una precisión de 56 % y recall de 70%. Comparando los resultados obtenidos con los de la literatura actual, este resultado parece muy prometedor. El volumen de eventos extraídos, lo consideramos muy bueno, sin embargo donde se puede mejorar es en la precisión.

Una forma de mejorar el rendimiento de la extracción es resolver mejor las correferencias. Para éste proyecto se adoptó un modelo muy sencillo de resolución de correferencias, el cual sustituye únicamente los pronombres relativos. Sin embargo, el otro tipo de pronombre, que es el personal, el cual es muy utilizado en el idioma español, no existe una regla simple para poder resolverlo, por lo que no tomamos acción alguna sobre estos.

Otra problema encontrado se presentó al querer aumentar ya sea el sujeto o el objeto a eventos dentro de la misma oración. Una forma de constatar si un evento compuesto por un determinado verbo necesita de un argumento, es verificar su valencia. No existe o no encontramos un recurso léxico que nos retorne esta información. Lo más cercano a lo que llegamos fue AnCora Verbs, un recurso léxico que contiene la categoría semántica de cada verbo y sus diátesis. Pero para empezar no están todos los verbos y aunque estuviesen, no es simple deducir si un verbo requiere o no de un argumento.

Por otro lado, la clusterización de los eventos presentó un coeficiente de Silhouette de 0,01, una pureza de 0,57 y una entropía de 1,41. Utilizando de referencia el artículo ”Unsupervised Techniques for Extracting and Clustering Complex Events in News” (Rusu, D., Hodson, J. and Kimball, A., 2014) se aprecia que el coeficiente de Silhouette reportado por ellos varía entre -0,17 y 0,42. Por lo que a priori nuestro resultado no es aceptable. Las demás métricas

no se evaluaron para ese artículo. Un problema encontrado con el clustering es la eficiencia del algoritmo. Para un único diario los tiempos superaron las 12hs y el tiempo de clusterización aumentaba de manera cuadrática con los datos.

Finalmente, el módulo de OCR juega el rol más importante si se quieren extraer los eventos, ya que extraer información de un texto ilegible es una tarea imposible. En su mayoría, la calidad de los diarios del siglo XIX es mala para los OCR que hay hoy en día, lo que hace que la solución final del proceso se degrade estrepitosamente. Si bien hicimos un corrector para intentar solucionar los errores cometidos por el OCR (entre otros tipos de errores), éste no puede corregirlos a todos, lo que hace que al extractor de eventos le lleguen oraciones erróneas.

9.2. Trabajo futuro

En cuanto al corrector del OCR, hay un par de cosas que se podrían hacer para mejorar la performance. Una es utilizar clases de equivalencia. La clase de equivalencia de un carácter es un conjunto de caracteres que son gráficamente similares a él y que por ende es común que el corrector los confunda (por ejemplo ‘e’ y ‘c’) estarían en la misma clase de equivalencia. Imponiendo que el corrector sólo pueda cambiar caracteres incorrectos otro de la misma clase de equivalencia debería mejorar bastante el resultado del mismo.

Otra cosa que podemos hacer es limitar nuestro diccionario del español. Como dijimos en la sección 5.3.2. Autocorrección de los documentos, utilizamos la herramienta Hunspell como diccionario del español, y solo corregimos una palabra si no está en el mismo. El problema es que este diccionario es tan completo, que muchas palabras mal escritas las termina encontrando como válidas lo cual termina siendo un detrimento para el resultado final.

Como mencionamos, la mala performance del clustering hacía que sea una tarea demasiada extensa si se tenían muchos eventos. La performance está directamente relacionada a nuestra medida de distancia entre eventos por lo

que, para un trabajo futuro, se recomienda probar con otra medida de distancia que sea más eficiente. Por ejemplo se podría no hacer todas las combinaciones posibles como hace nuestra medida de distancia a la hora de comparar (la medida de distancia se explica con detalle en la sección 7.1). Cabe mencionar que nosotros probamos un enfoque que no realizaba todas las combinaciones que si bien mejoró la performance, disminuyó la calidad de la misma y por ende la calidad del clustering.

Con respecto al etiquetado de clusters, se puede probar otro tipo de método para asignar etiquetas. Se pueden utilizar técnicas de doc2vec (Le & Mikolov 2014) para vectorizar cada sección o categoría. Luego, por otro lado, se calcula el vector centroide de los clusters (el cual puede no ser un vector que pertenezca al cluster) y se compara su similitud utilizando la distancia coseno.

Otro tipo de solución que podría ser muy interesante probar para la extracción de eventos, es la de utilizar aprendizaje supervisado, por ejemplo aprovechando el corpus ACE. En la sección de estado del arte, se mencionaron varias técnicas utilizadas para esto. Muchas de las soluciones descritas en dicha sección resolvían por ejemplo anáforas, las cuales en nuestra solución no están tenidas en cuenta, ya que nosotros hacemos extracción de eventos de manera local a la oración.

10. Glosario

Análisis morfológico: reconocer una palabra y construir una representación estructurada de ella, tomando en cuenta sus morfemas. A cada palabra se le asigna su lema y todas las categorías gramaticales (PoS tags) posibles.

Análisis sintáctico (parsing): Consiste en generar un árbol de análisis sintáctico (árbol de parsing) a partir de un texto y una gramática dados, si es posible. Existen distintos tipos de análisis sintácticos, entre los que se destacan full parsing, shallow parsing y dependency parsing. El full parsing provee un análisis de la estructura de la oración lo más detallado posible. El dependency parsing se basa en las relaciones de dependencia entre las palabras (como ser las funciones sintácticas), no existiendo el concepto de constituyentes.

Categoría léxica: paradigmas que forman las palabras en función de sus propiedades combinatorias y de sus rasgos formales. Ejemplos de estas: adjetivos, adverbios, conjunciones, determinantes, nombres, preposiciones, pronombres, verbos.

Corpus: Colección de material lingüístico. Para construir un corpus se deben definir las características deseadas: si se trata de material escrito u oral, cantidad de idiomas que presenta (monolingüe vs. multilingüe), el tipo de texto que se trata (por ej.: prensa, literario, científico), el dominio abarcado (por ej.: arte, lingüística), y si se encuentra anotado o no. Un corpus anotado es aquel que cuenta con anotaciones que lo enriquecen lingüísticamente. Estas surgen a partir de un proceso de análisis manual o automático. Las anotaciones pueden ser de distintos niveles: morfológicas, sintácticas, léxico-semánticas, entre otros.

Correferencias: La resolución de correferencias (o relaciones anafóricas) refiere a encontrar a qué entidad, aparecida anteriormente en el texto, se está haciendo referencia con una palabra o expresión en particular. Por ejemplo al decir “Juan jugaba a las bolitas y se le fueron a la calle”, vemos que “le” señala implícitamente a Juan previamente nombrado.

Entropía (clustering): La entropía se usa en los algoritmos de clustering como medida de la diversidad o heterogeneidad de un conjunto de objetos. Por ejemplo, la entropía en un cluster mide las proporciones de tipos de instancia que se encuentran en él, para así saber qué tan diverso o heterogéneo es.

Hiponimia: Es una relación de herencia entre palabras. De esta relación, participan hipónimo y un hiperónimo. El hipónimo es la palabra más específica y el hiperónimo la palabra más general, es decir que engloba a ésta. El hipónimo posee todos los rasgos semánticos del hiperónimo, pero además añade otras características semánticas que la diferencian de ésta.

Lema: Unidad autónoma constituyente del léxico de un idioma. Es una serie de caracteres que forman una unidad semántica y que puede constituir una entrada de diccionario. Por ejemplo el lema de “gatita” es “gato”, y el del verbo “corro” es “correr”.

Lenguaje natural: Lenguaje hablado por las personas (cualquiera de los idiomas del mundo se considera lenguaje natural, en contraste con los lenguajes formales definidos en computación).

Medida-F: Es una medida ponderada de las medidas precisión y recall.

Pos Tagging: Es el proceso de asociar una palabra a una determinada categoría léxica. Este proceso necesita de un análisis morfológico previo y del contexto de una palabra para determinar la etiqueta correcta, desambiguando entre todas las posibilidades.

Precisión: Es la fracción de instancias que son relevantes sobre las instancias que fueron recuperadas.

Pureza: Mide el porcentaje de datos bien clusterizados (que están en el cluster correcto) del sistema. Por ejemplo, dado un cluster, mide la proporción de la mayoría de instancias que pertenecen al mismo tipo con respecto al total de instancias de ese cluster.

Recall: Es la fracción de instancias que fueron recuperadas sobre la cantidad total de instancias relevantes.

Rouge-1, Rouge-2: Cuenta la cantidad de solapamiento de frases de palabras (1 palabra y 2 palabras respectivamente) que ocurre sobre dos textos.

Rouge-l: Esta medida cuenta el ratio de la cantidad de palabras que hay en la subsecuencia común más larga entre dos textos sobre la cantidad total de palabras. Este resultado permite saber qué tan ordenado están las palabras en el texto.

Silhouette Coefficient: Medida que usa para para evaluar un algoritmo de clustering. Mide la similitud entre los datos y el resto de datos en su cluster y lo compara con la similitud en el dato y los datos del cluster más cercano.

Sintagma: Es un tipo de constituyente sintáctico formado por un grupo de palabras que forman otros subconstituyentes, al menos uno de los cuales es un núcleo sintáctico.

Stop words: Palabras que se descartan de un texto, porque son consideradas irrelevantes. Son irrelevantes porque se trata de palabras que son muy frecuentes, por lo que no aportan información sustancial.

Word Embeddings: Es el nombre de un conjunto de técnicas utilizadas en procesamiento de lenguaje natural mediante las cuales las palabras o frases del vocabulario son mapeadas a vectores de números reales.

WordNet: Es una base de datos léxica, la cual tiene entre otras cosas una ontología o taxonomía entre palabras.

11. Bibliografía

Abby OCR. Available at: https://www.abbyy.com/es-la/finereader/what-is-ocr/ [Accessed December 11, 2018].

AnCora | Corpus. Available at: http://clic.ub.edu/corpus/es/ancora [Accessed December 11, 2018].

AnCora|Estructura argumentos ,AnCora 2.0: Argument Structure Guidelines for Catalan and Spanish. Available at: http://clic.ub.edu/corpus/webfm_send/52 [Accessed December 11, 2018].

Ahn, D., 2006, July. The stages of event extraction. In Proceedings of the Workshop on Annotating and Reasoning about Time and Events (pp. 1-8). Association for Computational Linguistics. https://dl.acm.org/citation.cfm?id=1629236 [Accessed December 13, 2018].

Ahonen-Myka, H., 2002. Discovery of frequent word sequences in text. In Pattern Detection and Discovery (pp. 180-189). Springer, Berlin, Heidelberg. https://www.researchgate.net/publication/2530992_Discovery_of_Frequent_Word_Sequences_in_Text [Accessed December 13, 2018].

Análisis semántico de lenguaje natural (ASLN). Facultad de Ingeniería, Uruguay. Available at: https://eva.fing.edu.uy/pluginfile.php/143120/mod_resource/content/2/Extraccion%20Abierta%20de%20Informacion.pdf [Accessed December 13, 2018].

Anónimo, ACE | Linguistic Data Consortium. Available at: https://www.ldc.upenn.edu/collaborations/past-projects/ace [Accessed December 13, 2018].

Biemann, C. 2006, Chinese whispers: an efficient graph clustering algorithm and its application to natural language processing problems. In Proceedings of the first workshop on graph based methods for natural language processing (pp. 73-80). Association for Computational Linguistics.

Chin-Yew Lin, 2004, ROUGE: A Package for Automatic Evaluation of Summaries. http://users.dsic.upv.es/~dpinto/duc/RougeLin.pdf [Accessed December 11, 2018]

Comparison of optical character recognition software - Wikipedia. Available at:https://en.wikipedia.org/wiki/Comparison_of_optical_character_recognition_software [Accessed December 13, 2018].

Cowie, J. and Lehnert, W., 1996. Information extraction. Communications of

the ACM, 39(1), pp.80-91. https://dl.acm.org/citation.cfm?id=234209 [Accessed December 11, 2018].

Fader, A., Soderland, S. and Etzioni, O., 2011, July. Identifying relations for open information extraction. In Proceedings of the conference on empirical methods in natural language processing (pp. 1535-1545). Association for Computational Linguistics. http://www.aclweb.org/anthology/D11-1142 [Accessed December 11, 2018].

Gamallo, P. and Garcia, M., 2015, September. Multilingual open information extraction. In Portuguese Conference on Artificial Intelligence (pp. 711-722). Springer, Cham. https://gramatica.usc.es/~gamallo/artigos-web/EPIA2015.pdf . [Accessed December 11, 2018].

Gramáticas formales para el lenguaje natural , 2017. Facultad de ingeniería,Uruguay. Available at: https://eva.fing.edu.uy/pluginfile.php/163636/mod_resource/content/0/GFLN2017-05_dep1.pdf [Accessed December 11, 2018].

Hearst, M.A., 1992, August. Automatic acquisition of hyponyms from large text corpora. In Proceedings of the 14th conference on Computational linguistics-Volume 2 (pp. 539-545). Association for Computational Linguistics. http://www.aclweb.org/anthology/C92-2082 [Accessed December 11, 2018].

Heliński, M., Kmieciak, M. and Parkoła, T., 2012. Report on the comparison of Tesseract and ABBYY FineReader OCR engines. http://lib.psnc.pl/dlibra/docmetadata?id=358&from=pubindex&dirids=1&lp=261 [Accessed December 11, 2018].

Jurafsky, D. and Martin, J., 2018 “Speech and Language Processing”. Available at: https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf [Accessed December 11, 2018].

Leacock, C., & Chodorow, M, 1998. Combining local context and WordNet similarity for word sense identification. WordNet: An electronic lexical database, 49(2), 265-283.

Le, Q. and Mikolov, T., 2014, January. Distributed representations of sentences and documents. In International Conference on Machine Learning (pp. 1188-1196). http://arxiv.org/abs/1405.4053 [Accessed December 11, 2018].

Mendoza, R.M.O. and Computacionales, C., 2007. Descubrimiento Automático de Hipónimos a partir de Texto no Estructurado (Doctoral dissertation, Master’s thesis, Instituto Nacional de Astrofísica, Óptica y Electrónica, México). https://ccc.inaoep.mx/~mmontesg/tesis%20estudiantes/TesisMaestria-Rosa

Ortega.pdf [Accessed December 11, 2018].

Pablo A. Paliza,2016, Corrección automática de errores de OCR en documentos semi-estructurados Available at: https://rdu.unc.edu.ar/bitstream/handle/11086/5587/Paliza.pdf?sequence=1 [Accessed December 11, 2018].

Rendón, E., Abundez, I., Arizmendi, A. and Quiroz, E.M., 2011. Internal versus external cluster validation indexes. International Journal of computers and communications, 5(1), pp.27-34. http://www.universitypress.org.uk/journals/cc/20-463.pdf [Accessed December 11, 2018].

Rusu, D., Hodson, J. and Kimball, A., 2014. Unsupervised techniques for extracting and clustering complex events in news. In Proceedings of the Second Workshop on EVENTS: Definition, Detection, Coreference, and Representation (pp. 26-34). http://www.aclweb.org/anthology/W14-2905 [Accessed December 11, 2018].

Tanner, S., 2004. Deciding whether optical character recognition is feasible. King's Digital Consultancy Services. https://www.kdl.kcl.ac.uk/fileadmin/documents/digifutures/2005_draft/END_Optical%20Character%20Recognition%20Feasibility_final.doc [Accessed December 11, 2018].

Yang, B. and Mitchell, T., 2016. Joint extraction of events and entities within a document context. arXiv preprint arXiv:1609.03632. https://www.cs.cmu.edu/~bishan/papers/joint_event_naacl16.pdf [Accessed December 11, 2018].

Zhila, A. and Gelbukh, A., 2014. Open Information Extraction for Spanish language based on syntactic constraints. In Proceedings of the ACL 2014 Student Research Workshop (pp. 78-85). http://www.aclweb.org/anthology/P14-3011 [Accessed December 11, 2018].