Determinación de la factividad de los eventos mencionados en el texto

Informe de Proyecto de Grado

Virginia Fernández

Esteban Fernández

Tutores:

Dra. Ing. Dina Wonsever
Dra. Ing. Aiala Rosá

Facultad de Ingeniería
Universidad de la República

Proyecto de Grado 2013

Resumen

La factividad es una propiedad de las referencias a eventos en textos. Un evento puede presentarse como ocurrido u ocurriendo, como no ocurrido u ocurriendo o como de ocurrencia eventual. Para determinarla es necesario observar el contexto donde dicho evento puede estar afectado por elementos de polaridad negativa, por operadores modales, o por predicados que afecten su valor de verdad, o combinaciones de todos estos.

Este proyecto aborda el desarrollo de una herramienta cuyo fin es determinar la factividad de un evento mencionado en textos en español utilizando métodos de aprendizaje automático. Para el análisis se cuenta con un corpus anotado por dos estudiantes de lingüística con una sintaxis de anotación llamada SIBILIA, la misma contiene información pertinente sobre el evento y la factividad del mismo. Dado que este corpus era muy pequeño se decidió expandirlo utilizando una primera versión del sistema que etiquetó nuevos textos de noticias y luego fue corregido manualmente por las tutoras.

Se toma como base numerosos trabajos previos hechos para diferentes idiomas, entre ellos el inglés y el croata.

El corpus mencionado anteriormente se utilizó para entrenar y testear un clasificador basado en los modelos *Conditional Random Fields* (CRF) y *Support Vector Machine* (SVM) ya que los mismos han obtenido muy buenos resultados en el área del procesamiento de lenguaje natural. También se realizó una prueba con árboles de decisión sin lograr resultados apropiados.

Los resultados obtenidos para los toolkits CRF++ y Yamcha son de 85,1% y 87,4% de Medida F respectivamente, estos superan ampliamente el valor de la línea base (68,5%) aunque todavía queda lugar para mejorar dado que no alcanzan a la línea tope (90,4%). La línea base consiste en un sistema simple basado en reglas que determina la factividad de los eventos en el texto. La línea tope, por otro lado, muestra la concordancia que existe entre las dos estudiantes de lingüística que anotaron manualmente el corpus.

Índice General

Capítulo 1

Introducción

En general, para cualquier tarea de procesamiento de lenguaje natural (PLN) que involucre cierto grado de comprensión de texto es importante reconocer la factividad de los eventos del texto.

La factividad de los eventos es la propiedad de un evento de presentarse como ocurrido o no. Esta no es un dato evidente, para determinarla es necesario hacer inferencias textuales, identificando elementos del contexto para poder deducir su valor. Esto se observa en los siguientes ejemplos donde se menciona un evento *explosión*, el cual es presentado como ocurrido (1.1), no ocurrido (1.2) e indefinido (1.3), es decir, no se puede afirmar si ocurrió o no.

(1.1) *La **explosión** tuvo lugar en la madrugada del jueves.*
(1.2) *El encargado de seguridad pudo evitar la **explosión**.*
(1.3) *Se esperaba que la **explosión** tuviera lugar dos días después.*

En estos ejemplos se observa como en (1.1) el contexto (tuvo lugar) indica que el evento es presentado como realizado, en (1.2) la palabra *evitar* presenta el evento como no realizado y en (1.3) *se esperaba* no permite identificar el evento ni como realizado ni como no realizado (no se sabe qué ocurrió con la explosión).

En general, los eventos están mencionados desde la perspectiva del autor, aunque pueden ser mencionados por varias fuentes (recurso muy usado en textos periodísticos), que pueden tener distintas perspectivas sobre la factividad del evento. Por esto, la factividad está muy relacionada con quién menciona el evento.

En [WMR09], artículo de referencia para este proyecto, se define a la factividad como "la actitud del enunciador (habitualmente el autor) respecto a la ocurrencia del evento referido".

La razón por la cual esta información es relevante, por ejemplo, en sistemas de QA[1], es que si estos no tuvieran en cuenta la factividad de los eventos llevaría a reportar información errónea. Un ejemplo de esto es mencionado en [SVP06b] donde se plantea que un sistema de QA que no tome en cuenta la factividad de (1.4) llevaría a responder erróneamente (1.6) a la pregunta (1.5), esta respuesta es errónea dado que el verbo *intentar* delante del evento *escalar* indica una posibilidad (no certeza).

(1.4) *George Mallory y Andrew Irvine intentaron escalar por primera vez el Everest en 1924.*
(1.5) ¿Cuándo escalaron el Everest por primera vez George Mallory y Andrew Irvine?
(1.6) En 1924.

Por otro lado, en el área de PLN, distinguir entre lo que se informó como un hecho frente a una posibilidad es una capacidad crucial para cualquier herramienta robusta de extracción de

[1] Los sistemas de QA (*Question Answering*) son sistemas que se encargan de responder de forma automática a las preguntas formuladas por un ser humano en lenguaje natural.

información. Un ejemplo de esto, tomado del ámbito biomédico, mencionado en [MB07], sería un sistema destinado a identificar y extraer interacciones entre entidades genéticas. Mientras (1.7) muestra una evidencia clara de una interacción de este tipo, el caso (1.8) provee solo una evidencia débil de esta interacción, marcado por el uso de *sugieren* y *puede*.

(1.7) *Nuestros resultados prueban que XfK89 inhibe Felin-9.*
(1.8) *Nuestros resultados* ***sugieren*** *que XfK89* ***puede*** *inhibir Felin-9.*

En este proyecto nos proponemos desarrollar una herramienta, basada en técnicas de aprendizaje automático, que determine la factividad de los eventos mencionados en un texto.

El aprendizaje automático es un campo de la inteligencia artificial que tiene como fin desarrollar técnicas que brindan a las computadoras la capacidad de aprender sin tener que programarlas, se considera un proceso en el cual se infiere conocimiento.

Con el objetivo de realizar un sistema de aprendizaje automático para inferir la factividad de los eventos, utilizamos la técnica de aprendizaje supervisado. El aprendizaje supervisado es una técnica utilizada para deducir una función a partir de un conjunto de entrenamiento dado, este conjunto por lo general está formado por un vector con los datos de entrada y los resultados que se espera obtener. El objetivo de dicha función es predecir el valor de salida para cualquier entrada válida, habiendo previamente analizado el conjunto de entrenamiento.

Se cuenta con un corpus en el cual los eventos tienen anotado manualmente el valor de factividad correspondiente. Esta información, junto con algunos atributos obtenidos de forma automática, es utilizada para el aprendizaje. El objetivo es generar un clasificador que, dado un texto, le asigne a cada evento el valor de factividad.

Dado que para nuestro proyecto la salida esperada para dicha función es un valor de factividad, nuestro problema de aprendizaje automático es un problema de clasificación.

Las ventajas que presentan los sistemas basados en aprendizaje automático frente a los basados en reglas es que el costo de construirlos es menor, se vuelven más manejables y, si el idioma cambiara, bastaría con introducir entrenamiento sobre un nuevo corpus sin necesidad de modificar el algoritmo. La principal desventaja es que requiere de muchos datos para entrenarlos de forma aceptable, muchas veces anotados manualmente, los cuales son difíciles de conseguir.

Estructura del documento

El documento se organiza de la siguiente forma:

El capítulo 2 muestra el marco del proyecto.

En el capítulo 3 se realiza un estudio del estado del arte, viendo los resultados y las técnicas aplicadas por otros trabajos.

El capítulo 4 trata del estudio teórico de la factividad de los eventos. Se muestran aquí las distintas propiedades que ayudan a determinar la factividad de un evento.

El capítulo 5 muestra el análisis del corpus y se presentan los datos estadísticos extraídos del mismo. También se presentan las líneas base y tope utilizadas, junto con las métricas que se utilizan para la evaluación de los sistemas. Por último se explica cómo se extendió el corpus para obtener más datos para el entrenamiento.

El capítulo 6 presenta la solución propuesta, donde se muestran los clasificadores elegidos y la determinación de los atributos. Estos atributos serán ajustados luego de testearlos contra el corpus de desarrollo.

El capítulo 7 muestra la implementación de los sistemas: línea base y sistema final. También se muestran las herramientas utilizadas (Freeling, Maltparser) y los *toolkits* utilizados (CRF++ y Yamcha).

En el capítulo 8 se presentan los experimentos realizados sobre el corpus de desarrollo, el proceso de selección y análisis de los atributos y el sistema final. En este caso se comparan los resultados de CRF++ y Yamcha y se presentan las curvas de aprendizaje.

Por último el capítulo 9 muestra las conclusiones finales, trabajos futuros y los aportes a la formación de los estudiantes.

Capítulo 2

Marco de Trabajo

En este capítulo se describe el contexto en el cual está situado el proyecto, observando que el mismo se encuentra dentro de un proyecto de análisis de textos más grande, llamado Temantex. Se muestran las herramientas disponibles en el marco de este proyecto y, en particular, SIBILA (el esquema de anotación utilizado para definir los eventos y sus atributos, dentro de los cuales se encuentra la factividad).

2.1. Proyecto Temantex

El proyecto se ubica dentro del Proyecto TEMANTEX, que el grupo de Procesamiento de Lenguaje Natural (PLN) viene llevando adelante desde hace varios años. El objetivo final de dicho proyecto es poder ordenar temporalmente los eventos de un texto. Para esto se debe determinar los eventos de un texto junto a sus atributos (entre los cuales se encuentra la factividad) y las expresiones temporales.

Se observa que determinar la factividad de los eventos es fundamental para poder ordenarlos temporalmente, ya que no tiene sentido ordenar temporalmente un evento que no sucedió.

Contamos con varias etapas de este proyecto que serán fundamentales para el nuestro:

1. Definición de un esquema de anotación para eventos (SIBILA).

2. Corpus anotado con los eventos y sus atributos siguiendo dicho esquema. Dicha anotación fue realizada por dos estudiantes de Lingüística.

3. Listas de verbos y otros recursos léxicos que ayudan a determinar el valor de factividad de los eventos.

4. Informe de un proyecto de grado que determina los eventos de un texto [AD10] junto con algunos de sus atributos.

2.2. SIBILA

SIBILA es un esquema de anotación para eventos basado en el propuesto por TimeML [PCI+03]. La diferencia fundamental entre ambos esquemas, para nuestro trabajo, es que SIBILA determina la factividad de los eventos, pero TimeML no lo hace. En el esquema SIBILA se propone un atributo **FACTIVIDAD** que indica si los eventos ocurrieron, no ocurrieron o no se sabe si ocurrieron.
Se definen a continuación las etiquetas de SIBILA que consideramos útiles para la determinación de la factividad. Una explicación completa de estas etiquetas se encuentra en [WMR08].

2.2.1. Etiqueta Evento

Según la anotación de SIBILA un evento es "cualquier tipo de situación o acontecimiento denotado por un predicado". Estos se dividen en:

- **Acciones:** acontecimientos llevados a cabo voluntariamente por un sujeto agente.
  - *Los antropólogos forenses **delimitaron** el predio*.
- **Procesos**: acontecimientos desencadenados espontáneamente o causados por una fuerza externa al proceso.
  - *Los árboles **están floreciendo** prematuramente por las altas temperaturas*.
  - *Se prevé que los fuertes vientos **derrumbarán** varios techos*.
- **Estados:** situaciones que se mantienen a lo largo de un período o son permanentes.
  - *El tránsito **está detenido** a causa de los cortes de ruta.*

La etiqueta evento contiene varios atributos, entre los cuales se encuentran: la categoría gramatical, la forma verbal, el modo, el tiempo, la polaridad, la modalidad, la clase y la factividad. El atributo más importante para este proyecto es la factividad del evento, ya que la herramienta desarrollada será la encargada de asignar un valor para este atributo. Esta representa el grado de certeza del enunciador, respecto a la ocurrencia del evento referido. Puede tomar los valores: REALIZADO, NO_REALIZADO, FUTURO_PROG, FUTURO_NEG, POSIBLE o INDEF. A lo largo del informe se profundizará sobre la definición de este atributo y sus posibles valores.

Dado que para determinar la factividad de un evento es necesario considerar muchas características del evento y del contexto en el cual se encuentra, varios de los otros atributos ayudan a la hora de determinar dicha factividad, estos son:

modo: Los diferentes modos afectan la factividad de un evento, por ejemplo en (2.1) se observa que el verbo *trabajar*, que está en modo indicativo, expresa un hecho real (sucedió), en (2.2) en cambio, el verbo *recitar* está en subjuntivo (expresa deseo o expectativa) y no permite saber si el evento (*recitara un poema*) ocurrió, haciéndolo indefinido. En (2.3) a su vez el verbo *caminar* está en imperativo (usado para expresar una orden) y toma el valor de indefinido ya que no sabemos si el evento (caminen) ocurrió o no.

(2.1) *Luis **trabajó** en un banco.*
(2.2) *El profesor quería que Juan **recitara** un poema.*
(2.3) **Caminen** por esa calle.

tiempo: El tiempo ayuda a determinar la factividad, por ejemplo en (2.4) el evento *jugarán* está en tiempo futuro lo cual no permite saber si dicho evento realmente sucedió o no, pero está programado a pasar en el futuro, por lo que toma el valor de futuro programado. En cambio en (2.5) el evento *marcó* se encuentra en pasado, lo cual indica que el mismo sucedió.

(2.4) *Solo el Tata González y Diego Rolan **jugarán** en el arranque del segundo torneo europeo.*
(2.5) *Luis Suárez **marcó** el tercer gol del encuentro.*

polaridad: La polaridad afecta directamente la factividad de un evento, esto se muestra en (2.6) donde la palabra *no* afecta el evento *jugó al fútbol* el cual toma el valor de no realizado.

(2.6) *Álvaro Recoba no **jugó al fútbol** este jueves.*

modalidad: La modalidad ayuda a determinar el evento, generalmente indicando posibilidad en lugar de certeza sobre el evento referido, por ejemplo en (2.7), la palabra *posible* indica incertidumbre sobre el evento *paro*.

(2.7) *Hay inquietud en el gobierno por el posible **paro** de la semana próxima.*

2.2.2. Etiqueta Índice

Esta etiqueta señala los elementos del texto que ayudan a determinar los atributos de la etiqueta evento. Contiene un atributo llamado **clase** cuyo valor depende del valor del atributo de la etiqueta evento que ayude a determinar. Los posibles valores para el atributo clase que inciden en el valor del atributo factividad son: modalidad, factividad, polaridad y condición. Los ejemplos de la sección anterior muestran subrayados ejemplos de los índices de modalidad (*posible* en 2.7) y polaridad (*no* en 2.6). Algunos ejemplos de índices de clase condición son: *si*, *a condición de que*, *en caso de que*, *siempre y cuando*, *con tal de que*. Un ejemplo (*inminente*) de un índice con clase factividad se muestra en (2.8), donde dicho índice determina que el evento *traslado* sea algo programado a pasar en el futuro, tomando el valor de factividad FUT_PROG.

(2.8) *En este marco se realizó ayer una reunión entre la Ministra y los jefes de Policía del país, en la que se anunció el inminente **traslado** de presos del Compen al interior del país.*

Capítulo 3

Antecedentes

En este capítulo se presentan un conjunto de trabajos que se utilizaron para analizar el estado del arte y como base para la implementación del proyecto. Estos son recientes, y en su mayoría basados en reglas para el idioma inglés. También se cuenta con un proyecto basado en métodos de aprendizaje automático para el idioma croata. Por último se presenta un resumen del proyecto de grado que determina los eventos en el texto. De aquellos trabajos que nos fueron más útiles realizamos un resumen en el Anexo.

3.1. Trabajos sobre la Factividad

En "*Computing relative polarity for textual inference*" [NCK06] se determina el valor de factividad de los argumentos de los predicados implicativos (*lograr*, *forzar*, etc.), estos son los que inciden en el valor de verdad de sus argumentos proposicionales [KAR70]. Para esto se consideran las propiedades de los predicados (sólo considerando los verbos) y el contexto de polaridad del evento.

Los predicados implicativos pueden estar anidados, por lo que la polaridad del contexto depende de las cadenas de predicados y de los operadores que contenga la oración en que se encuentra. Por todo esto, este trabajo propone un algoritmo que calcula la polaridad para ver cómo afectan los predicados implicativos y los modificadores de polaridad en la factividad de un evento que está dentro de su alcance.

Si bien para nuestro proyecto no se utiliza dicho algoritmo, se buscó la forma de incluir este concepto a la hora de seleccionar los atributos necesarios para el aprendizaje, incluyendo atributos que tienen en cuenta los predicados implicativos y la polaridad.

En SlinkET [SVP06a] por otro lado, se propone un analizador para la identificación del grado de certeza acerca de la ocurrencia de los eventos presentes en el texto, basado en TimeML [TIM08, SAU08] en el contexto del *framework* Tarsqui [TAR08] (*Temporal Awareness and Reasoning Systems for Question Interpretation*). El objetivo de SlinkET es generar *links* de tipo SLINK (vínculo de subordinación entre dos eventos) y determinar para cada uno de estos la certeza que se tiene sobre su ocurrencia.

Los posibles valores de factividad son:

Fáctico: cuando el evento se presenta como ocurrido.
Contra fáctico: cuando el evento se presenta como no ocurrido.

Evidencial: cuando el evento es reportado por alguien, quien lo menciona como ocurrido.
Evidencial negativo: cuando el evento es reportado por alguien como no ocurrido.
Modal: si el evento es mencionado como posible.
Condicional: eventos anotados como construcciones condicionales.

El valor modal de este trabajo fue tomado en cuenta en nuestra solución incluyendo atributos que miden la posibilidad de los eventos. Los valores de "Evidencial" o "Evidencial negativo" fueron evaluados con la idea de agregar un atributo para los eventos de reporte el cual fue descartado ya que incluirlo en la lista de atributos implicaba un trabajo que excedía el alcance de del proyecto.

En De Facto (definido en [SP12] e implementado en [RS08]) se propone un modelo computacional lingüístico formado por un algoritmo que determina la factividad de los eventos. Se definen los Predicados de Selección de eventos (ESP) como los predicados con un argumento que denotan un evento (por ejemplo predicados de reporte, conocimiento, creencia). Estos ayudan a determinar la factividad del evento. Existen dos tipos de ESPs: SIPs (predicados que introducen fuentes) y NSIPs (predicados que no introducen fuentes).

Se cuenta con una lista de 11 indicadores de polaridad, entre ellos: *no*, *tampoco*, *ni*, *ninguno*, *nadie*, *nada*. También cuenta con 31 indicadores de factividad, entre ellos: *ciertamente*, *aparente*, *imposible*, *necesario*, *supuestamente*.

El modelo planteado en [SP12] es implementado en [RS08] (perfilador de factividad) y probado contra un corpus construido para la tarea, produciendo una Medida F entre 0,7 y 0,8.

En [RS08] se diseña y desarrolla un **perfilador de factividad**, es decir, una herramienta que determina la factividad de todos los eventos del texto. La factividad es definida en este trabajo como "la categoría a cargo del estado factual de los eventos". Es decir, marcan si un evento es presentado como fáctico (sucedió) o contra fáctico (no sucedió) o si su ocurrencia o no ocurrencia es incierta.

La factividad es representada por una escala de doble eje, siendo un eje el grado de compromiso del autor sobre la certeza de lo que se afirma, que va desde incierto hasta absolutamente seguro, y el de la polaridad, representándola, entonces como un par <modalidad, polaridad>. Los valores de modalidad son: cierto (CT), probable (PR), posible (PS) y desconocido (UN), mientras que la polaridad puede ser positiva (+) o negativa (-). Se observa que tanto la definición de factividad como la combinación de valores que se propone son similares a la propuesta por SIBILA. Donde <CT,+> equivale a **realizado**, <CT,-> a **no realizado,** <PS,+> a **posible**, <PR,+> y <PR,-> a **futuro programado** y **futuro negado** respectivamente y <UN, UN> a **indefinido**.

Cuando se menciona un evento, siempre existe una fuente que se compromete con la factividad de ese evento. Por esto, la factividad de los eventos es relativa a los participantes, referidos en este trabajo como **fuentes de factividad.**

Si bien determinar la factividad de un evento para cada fuente haría el trabajo más completo escapa de nuestros objetivos, ya que los mismos son determinar la factividad con la cual los eventos del texto son presentados, desde la perspectiva del autor. A su vez, existe un trabajo [AR11] que trata sobre la determinación de fuentes, el cual se detalla en el anexo en la sección A.

En ambos trabajos encontramos varios elementos que tenemos en cuenta. Nuestro trabajo plantea también una escala discreta, necesaria para poder categorizar la factividad. A su vez, también consideramos listas de palabras que indican negación y posibilidad.

Un trabajo que fue de gran interés es "*Learning event factuality in Croatian texts*" [GSB12], este tiene como objetivo identificar la certeza (nivel de confianza sobre la realización del evento) y polaridad (ocurrencia o no ocurrencia) de los eventos, así como determinar la viabilidad de este tipo de herramientas para las lenguas con pocos recursos. Este módulo sirve como base para la integración de aprendizaje automático (en particular supervisado), además de diferenciarse de los anteriores al determinar la factividad de eventos en textos croatas.

El módulo define la factividad como el par <certeza, polaridad> donde la certeza puede tomar los valores cierto, probable y posible, y la polaridad puede tomar los valores positivo y negativo. Nuevamente estos valores son similares a los propuestos por SIBILA, con la diferencia de que no hay un valor para la factividad indefinida.

Los atributos utilizados basados en el léxico son: *token*[2] y lema, descripción morfosintáctica del evento, sustantivo verbal, oración interrogativa y argumento de otro evento, entre otros. Para clasificar la polaridad se incluyeron atributos que utilizan listas de pistas negativas. Y para clasificar la certeza se utilizaron listas de pistas condicionales, listas de pistas en tiempo futuro y listas de pistas de posibilidad.

Tanto para las características de clasificación de la polaridad como de la certeza se toma alguna de las listas de pistas y se busca que ocurra alguna de las palabras o bien en el contexto izquierdo del evento, o bien en el derecho, también se busca la pista según la proximidad al evento ya sea inmediata o lejana.

Este es el único trabajo de referencia que utiliza métodos de aprendizaje automático, por lo cual se sacaron ideas de estos atributos. En particular se generaron atributos basados en la pertenencia a una lista de negación o una lista de posibilidad respectivamente. En nuestro caso se cuenta con herramientas que permiten realizar un análisis mayor del texto, por ejemplo obteniendo el árbol de dependencias[3] de la oración o determinando si una palabra está en futuro (sin necesidad de tener una lista de palabras en futuro). Por lo que la distancia del evento a alguna palabra de las listas de pistas y la búsqueda de pistas en los contextos puede ser cambiado por atributos que determinen si existe una palabra dentro del subárbol de dependencias que tenga como raíz el evento, o sea antecesor de dicho evento.

El aprendizaje sobre la clasificación de la polaridad y certeza se realiza utilizando *Support Vector Machine* (SVM). El corpus está formado por un conjunto de 4596 eventos. Un 78,6% de los eventos fueron anotados como positivos y seguros, algo esperado en los textos periodísticos. Sus resultados indican que si bien utilizar características basadas en el léxico para la polaridad produce buenos resultados, la predicción de la certeza requiero del uso de características más sofisticadas. Sin embargo, sus estadísticas dicen que estos resultados no son significativos comparados con una línea base simple basada en reglas.

[2] Por *token* se entiende a una cadena de caracteres (palabras, números y símbolos).
[3] Un árbol de dependencias es un grafo dirigido que representa las dependencias de las palabras de una oración. Las dependencias se establecen entre pares de palabras, donde una es principal y la otra está subordinada a (o dependiente de) la primera.

3.2. Reconocimiento automático de eventos en textos

El objetivo de este trabajo [AD10] es elaborar un sistema de reconocimiento de eventos para textos en idioma español basado en técnicas de aprendizaje automático. Dicho sistema será capaz de recibir como entrada texto plano en español y delimitar los eventos en el texto de entrada y los atributos que puedan ser recuperados por un analizador morfosintáctico (categoría, forma verbal y modo). Se define un evento como cualquier tipo de situación o acontecimiento que ocurre, o elementos que describen un estado o circunstancia. Los eventos pueden ser expresados por verbos conjugados o en infinitivo, predicados en general y frases preposicionales.

Se utilizaron los métodos de aprendizaje automático supervisado *Conditional Random Fields* (CRF) y *Support Vector Machines* (SVM). La evaluación de los clasificadores fue de una Medida F de 76,72% para CRF y 80,34% para SVM.

Este trabajo resulta de gran interés por varias razones:

1. Se encuentra en el mismo marco de trabajo que nuestro proyecto y utiliza el mismo corpus y esquema de anotación.

2. Utiliza métodos basados en aprendizaje automático que utilizamos (SVM y CRF), por lo que también uno de los mayores problemas que enfrentó fue la selección de atributos.

3. Fue útil al momento de decidir restringir los eventos solo a los verbos, ya que muestra que estos son la mayoría, y su identificación automática da resultados mucho mejores que para otras categorías.

4. Observando varios de los atributos seleccionados en este proyecto se decidió incluir algunos de ellos al nuestro, estos son los atributos básicos: *token*, lema, tiempo y modo.

Capítulo 4

La Factividad del evento

En este capítulo presentamos la definición de factividad de un evento y mostramos los posibles valores que esta puede tomar junto con ejemplos para cada valor.

Luego analizamos los distintos tipos de elementos que inciden en el valor de factividad de los eventos referidos en el texto. Estos son un resumen de los mencionados en [WMR09].

4.1. Definición de Factividad

En base a [WMR08] se define a la factividad como la certeza del enunciador sobre la ocurrencia del evento referido. El enunciador generalmente es el autor del texto, pero puede ser variable, y el autor puede no coincidir con el enunciador o puede no expresarse sobre la factividad del evento.

A su vez, la factividad está muy relacionada con la **modalidad epistémica**, es decir, la expresión del grado de certeza o duda que el emisor muestra con respecto a la verdad del evento referido en su enunciado (por ejemplo *Seguro que lo sabe*).

Inicialmente se proponen cinco valores de factividad: r**ealizado**, **no realizado**, **futuro programado**, **futuro negado** e **indefinido**. Luego, en el proceso de anotación, se agrega el valor **posible.**

Un evento con factividad **realizado** se presenta como ocurrido u ocurriendo, mientras que uno con factividad **no_realizado** se presenta como no ocurrido. Ambos se presentan con certeza por parte del enunciador.

El resto de los valores de factividad indican que la realización del evento es desconocida. Dentro de estos se encuentran los valores de futuro programado (**fut_prog**) y negado (**fut_neg**), que se considera que van o no van a pasar respectivamente en el futuro. Para los indefinidos (**indef**) no hay indicios de si pasaron o no, por lo general están en pasado, con lo cual su factividad no va a cambiar (ya pasó o no pasó). Por último **posible** indica que el evento podría pasar.

La definición de evento vista en la sección 2.2.1 implica tanto los verbos, como los adjetivos o nombres. A pesar de esto, como nuestro objetivo no es determinar los eventos, sino, dado los eventos determinar la factividad, se decidió simplificar el problema y focalizarnos solo en los verbos. Esto se debe a que, como se menciona en [AD10], la mayoría de los verbos son eventos, y la mayoría de los eventos son verbos. A su vez dicho trabajo ya tiene como objetivo determinar los eventos del texto.

A continuación se muestran algunos ejemplos de cada valor de factividad, donde los eventos se representan en negrita y luego se muestra la factividad de cada uno de ellos. Por lo dicho anteriormente, solo presentamos ejemplos de eventos que sean verbos.

(4.1) *Según sus datos, "**hay** miles de insumisos condenados que no **han solicitado** el indulto porque no **creen haber cometido** un delito, sino que **han ejercido** su derecho a la desobediencia civil".*

Evento	**Factividad**
hay	realizado
han solicitado	no_realizado
creen	no_realizado
haber cometido	indef
han ejercido	realizado

(4.2) *Ahora bien, la **empobrecida** y **semiparalizada** economía rusa podría **colapsar** definitivamente, **arrastrando** consigo a una población que durante un decenio **ha soportado** un permanente estado de abandono por parte de los sucesivos gobiernos y que **ha estado viviendo** al límite del hambre como en las peores épocas de la colectivización estaliniana.*

Evento	**Factividad**
empobrecida	realizado
semiparalizada	realizado
colapsar	posible
arrastrando	posible
ha soportado	realizado
ha estado viviendo	realizado

(4.3) *El "proyecto de protocolo de acuerdo" de nueve puntos, que los agentes sociales **volverán a abordar** el próximo lunes, **reafirma** la relación entre la "indemnización" de desempleo y la "ayuda"*

Evento	**Factividad**
volverán a abordar	fut_prog
reafirma	realizado

(4.4) *A pesar de los sufrimientos, ellos no **exigirán** su marcha*.

Evento	**Factividad**
exigirán	fut_neg

(4.5) *Sin embargo, los abogados de la empresa **indicaron** una vez más que **proyectan insistir** en las apelaciones para **impedir** que esos remedios "extremos y sin justificación" **se lleven a efecto**.*

Evento	**Factividad**
indicaron	realizado
proyectan	realizado
insistir	posible
impedir	posible
se lleven a efecto	fut_neg

(4.6) *El Komercni Banka (Banco Comercial), uno de los cuatro bancos más grandes de la República Checa, **anunció** hoy que **despedirá** a 2.300 empleados más antes de finales del año dentro del proceso de saneamiento de la entidad estatal.*

Evento	**Factividad**
anunció	realizado
despedirá	fut_prog

4.2. Elementos lingüísticos que inciden en la factividad

En esta sección se analizan los distintos tipos de elementos que inciden en el valor de factividad de los eventos referidos en el texto que fueron útiles para este proyecto. Estos son un resumen de los mencionados en [WMR09].

4.2.1. Modo Verbal

El modo verbal es un rasgo morfológico de los verbos, este representa las diversas formas en que la acción del verbo puede expresarse. En general, el modo verbal es sensible al hecho de que los estados de cosas se presenten como conocidos, imaginados, deseados, logrados, negados, etc.

Los predicados en **subjuntivo** presentan a los eventos como: una evaluación, una emoción, una intención o deseo, pero su factividad no es uniforme. El subjuntivo puede inducir eventos fácticos en el caso de los predicados valorativos o de afección (4.7 y 4.8), y el caso de los predicados realizativos (4.9), todos estos tomando el valor de realizado (siempre que no estén negados). Otros subjuntivos expresan en cambio duda, incertidumbre, posibilidad o sospecha, entre otros, tomando el valor de factividad indefinida o posible (4.10, 4.11, 4.12 y 4.13).

(4.7) *Lamento que **vinieras**.*
(4.8) *Fue necesario que **renunciaras**.*
(4.9) *Logró que **salvaran** el examen.*
(4.10) *Duda de que **lleguen** temprano.*
(4.11) *Temía que no **terminaran** a tiempo.*
(4.12) *Es posible que no **respondan**.*
(4.13) *La orden de que se **presentaran** al otro día.*

El modo **indicativo** se utiliza para describir el mundo de la realidad, por lo general indicando eventos como realizados o no realizados (certeza) (4.14)

(4.14) *Pedro (no) **estudia** todas las noches.*

El modo **imperativo** expresa una orden, solicitud o prohibición (4.15), generalmente indicando un evento como indefinido o posible.

(4.15) ***Estudiá** si no querés reprobar.*

El modo **condicional** se utiliza para expresar incertidumbre, particularmente (pero no exclusivamente) en oraciones condicionales, indicando generalmente el evento como indefinido o posible (4.17).

(4.17) *Si yo comiera más, **estaría** muy gordo.*

4.2.2. Tiempo Verbal

El tiempo verbal expresa el momento de la acción expresada por el verbo, este sirve para saber si una situación o evento es anterior, simultáneo o posterior al momento de la enunciación.

A grandes rasgos un evento presentado en tiempo pasado define un evento realizado, mientras que uno en tiempo futuro define a un evento como no realizado o incierto. Sin embargo, para el caso de los eventos históricos (4.18) las formas verbales como presente y el futuro de indicativo pueden ser utilizados en eventos realizados, ocurridos en el pasado.

(4.18) ***Abandona** el país en 1968, un año después se **instalará** definitivamente en Madrid.*

4.2.3. Polaridad negativa

La polaridad negativa influye en el valor que toma la factividad de un evento. En su forma más simple, un evento mencionado como **realizado** (4.19) es transformado en **no_realizado** (4.20) con un elemento de polaridad negativa.

(4.19) *Juan fue a la reunión.*
(4.20) *Juan **no** fue a la reunión*.

4.2.4. Presuposición e implicación

Para el caso de predicados que implican ocurrencia o no ocurrencia de su argumento es posible identificar dos grandes tipos de relaciones lógicas, las relaciones de presuposición y de implicación ([KIP70], [KAR70]). Las mismas se detallan a continuación.

Presuposición

Estos predicados establecen una relación de presuposición lógica con su argumento. Por ejemplo en (4.21) se presupone que hace frío, mientras que en (4.22) no se deduce el mismo tipo de presuposición. Un caso similar se da entre los ejemplos (4.23 y 4.24), ya que el predicado de (4.23) implica un estado mental del sujeto Juan pero la factividad del evento es indefinida, en cambio en (4.24) el predicado presupone el argumento como verdadero. También se observa que, tanto la negación (4.25) como los operadores modales (4.26) que afectan estos predicados, no influyen con el valor de factividad deducida del evento. Por último, un caso similar al anterior se da para la interrogación (4.27).

(4.21) *Es **asombroso** que haga frío.*
(4.22) *Es **posible** que haga frío.*
(4.23) *Juan está **seguro** de que María va a venir.*
(4.24) *Juan es **consciente** de que va a venir.*
(4.25) *Juan **no** es **consciente** de que María va a venir.*
(4.26) *Juan **sería consciente** de que María vendrá.*
(4.27) *¿Juan es **consciente** de que María va a venir?*

Implicación

Estos predicados establecen una relación de implicación lógica con su argumento. Esta puede ser positiva (*lograr*, *intentar*, etc.) o negativa (*evitar*, *rehusar*, etc.). La implicación permite deducir la verdad o falsedad de la proposición insertada a partir de la verdad o falsedad del predicado implicativo que rige la subordinación. La siguiente tabla muestra la polaridad de un evento dentro del alcance de un predicado implicativo, calculado según la polaridad del contexto (columnas) y el tipo de dicho predicado (filas). La lista completa se encuentra en el anexo en la sección B, la misma fue extraída de [MW13].

Predicado	Implicación	
	Polaridad del contexto	
	Polaridad Positiva (+)	**Polaridad Negativa (-)**
lograr **(+)(+)/(-)(-)**	positiva (+) *Juan **logró** abrir la puerta.* Juan abrió la puerta.	negativa (-) *Juan no **logró** abrir la puerta.* Juan no abrió la puerta.
olvidar(se) de + inf **(+)(-)/(-)(+)**	negativa (-) *Juan se **olvidó de** abrir la puerta.* Juan no abrió la puerta.	positiva(+) *Juan no se **olvidó de** abrir la puerta.* Juan abrió la puerta.
verse obligado a **(+)(+)**	positivo(+) *Juan **se vio obligado a** abrir la puerta.* Juan abrió la puerta.	ninguna *Juan no **se vio obligado a** abrir la puerta.* Juan puede o no haber abierto la puerta.
negarse **(+)(-)**	negativo (-) *Juan **se negó** a abrir la puerta.* Juan no abrió la puerta.	ninguna *Juan no **se negó** a abrir la puerta.* Juan puede o no haber abierto la puerta.
intentar **(-)(-)**	ninguna *Juan **intentó** abrir la puerta.* Juan puede o no haber abierto la puerta.	negativa (-) *Juan no **intentó** abrir la puerta.* Juan no abrió la puerta.
dudar en **(-)(+)**	ninguna *Juan **dudó en** abrir la puerta.* Juan puede o no haber abierto la puerta.	positivo (+) *Juan no **dudó en** abrir la puerta.* Juan abrió la puerta.

Tabla 1 - Tabla de implicaciones

4.2.5. Predicados introductores de hipótesis

Otro elemento léxico que influye en la factividad del argumento son los predicados que explícitamente señalan un valor hipotético en su argumento. Ejemplos de esta clase de verbos son *suponer*, *hipotetizar*, etc. Un ejemplo de predicado que introduce una hipótesis es (4.28) donde no se sabe con certeza el resultado del evento (no sabemos si el tren llegó o no), en cambio en (4.29) sí se sabe.

(4.28)***Supongamos** que el tren llegó a las 5.*
(4.29)*El tren llegó a las 4.*

4.2.6. Modalidades proposicionales

Las modalidades proposicionales refieren a la actitud del hablante con respecto a la posibilidad o necesidad de que ocurra un evento.

Los elementos que introducen modalidad son:

- adjetivos (*Es **posible** que llueva.*)
- adverbios (***Posiblemente** llueva.*)
- verbos en perífrasis (***Puede** llover.*)

Se observa que es muy importante la influencia de la modalidad sobre el valor de la factividad del evento, ya que es muy difícil concluir que el evento sea realizado.

La lista completa de predicados que señalan posibilidad se encuentra en el anexo en la sección B, esta lista fue extraída de [RS08].

Capítulo 5

Análisis del Corpus

En este capítulo se presenta el corpus anotado y los resultados obtenidos de su análisis. Se definen las líneas base y tope utilizando dicho corpus, junto con las métricas utilizadas para calcularlas. Estas métricas serán también utilizadas más adelante, en la evaluación del clasificador construido. Por último se explica el procedimiento seguido para extender el corpus disponible y las métricas del corpus total.

5.1. Métricas

5.1.1. Matriz de confusión

En métodos de aprendizaje automático se le llama matriz de confusión a una matriz que permite visualizar el desempeño del algoritmo.

Cada columna de la matriz representa las predicciones de una clase, mientras que cada fila representa la clase real.

Asumiendo que estamos en una clasificación binaria, donde una de las clases es + y la otra es -, se definen:

Verdadero positivo (*true positive*): cantidad de elementos de la clase + correctamente clasificados como +.

Verdaderos negativos (*true negative*): cantidad de elementos de la clase - correctamente clasificados como -.

Falsos positivos (*false positives*): cantidad de elementos de la clase - incorrectamente clasificados como +.

Falsos negativos (*false negatives*): cantidad de elementos de la clase + incorrectamente clasificados como -.

		Clase predicha	
		+	**-**
Clase real	**+**	Verdaderos Positivos (TP)	Falsos Negativos (FN)
	-	Falsos Positivos (FP)	Verdaderos Negativos (TN)

Tabla 2 - Matriz de confusión para clasificación binaria

Precisión (*precision*): es la probabilidad de que un ejemplo clasificado como + (elegido al azar) pertenezca realmente a esa clase. Es decir, $P = \frac{TP}{TP+FP}$

Recuperación (*recall*): es la probabilidad de que un elemento de clase + (elegido al azar) haya sido clasificado como +. Es decir, $R = \frac{TP}{TP+FN}$

Para un problema multi-clase, estos valores quedan:

Precisión: $P_i = \frac{m_{ii}}{\sum_j m_{ji}}$

Recuperación: $R_i = \frac{m_{ii}}{\sum_j m_{ij}}$

Donde m_{ij} es el valor de la matriz de confusión en la fila i y la columna j, es decir, la cantidad de elementos de la clase i clasificados como j.

Una precisión de 1 (medida perfecta) implica que todos los casos clasificados como + pertenecen a dicha clase. En cambio una recuperación de 1 implica que todos los elementos de la clase + fueron clasificados como +.

5.1.2. Medida F

Es una medida que combina precisión y recuperación, se define como la media armónica de ambos.

$$\mathrm{F}_1 = 2 * \frac{precision * recall}{precision + recall}$$

Este es un caso particular de la fórmula general (donde β es un real no negativo)

$$\mathrm{F}_\beta = (1 + \beta^2) * \frac{precision * recall}{\beta^2 * precision + recall}$$

El parámetro β indica cuánto más peso se le da a la precisión frente a la recuperación, en el caso de la Medida F β vale 1 por lo que ambos pesan igual.

5.1.3. Exactitud

La exactitud (*accuracy*) se refiere a cuán cerca del valor real se encuentra el valor medido. Se define la exactitud en una clasificación binaria como: $\frac{TP+TN}{TP+FP+TN+FN}$.

Donde en una clasificación multi-clase dicha definición queda:
$Exactitud = \frac{\sum_i m_{ii}}{\sum_i \sum_j m_{ij}}$ donde m_{ij} es el mismo que el usado para la precisión y recuperación.

5.2. El corpus anotado

Se denomina corpus a un conjunto lo más extenso y ordenado posible de datos o textos científicos, literarios, etc., que pueden servir de base a una investigación. Estos corpus son a menudo sometidos a un proceso de anotación. Este proceso consiste en introducir una serie de etiquetas que aportan información lingüística o declarativa al texto del corpus.

En el área de PLN es muy importante tener colecciones de datos (típicamente documentos) etiquetados indicando qué elementos aparecen en ellos. Es decir, tener documentos en que se describa, de forma que lo pueda entender una máquina, ciertos aspectos que aparecen en el texto. Estas etiquetas o anotaciones dependen de la tarea a la que estará destinado dicho corpus. El principal problema de estos corpus anotados es que requieren de un enorme trabajo manual para su construcción.

En nuestro caso, una de las tareas del proyecto Temantex consistió en la anotación manual de un corpus de textos en español por medio de dos estudiantes avanzados de Lingüística. Este corpus consiste en artículos de prensa y textos históricos, y el proceso de anotación fue realizado por ambas anotadoras mayormente de manera independiente siguiendo el esquema de anotación SIBILA.

5.2.1. Datos estadísticos

Anotadora	**Cantidad de eventos totales**	**Cantidad de oraciones**	**Cantidad de *tokens***
1	1067	223	6961
2	991	199	6347

Tabla 3 - Datos del corpus anotado

La Tabla 3 contiene los datos de la cantidad de eventos, oraciones y *tokens* que anotó cada anotadora en el corpus. Se observa que una de las estudiantes anotó más oraciones y por lo tanto más eventos, estas oraciones fueron en realidad anotados por ambas anotadoras en conjunto, dado que se estaban familiarizando con el esquema de anotación y debían hacerse consultas. El resto de las anotaciones fueron realizadas de manera independiente.

Dentro de estos eventos, las anotadoras coincidieron en 896, pero solo 875 de ellos tenían valor de factividad asignado por ambas anotadoras.

La Tabla 4 muestra los valores de factividad asignados a cada evento por cada anotadora, asociados a los 875 eventos anotados por ambas anotadoras que contenían valores de factividad asociado.

Valor de Factividad	Anotadora 1	Anotadora 2
Indef	99	63
No_realizado	47	48
Fut_neg	9	12
Realizado	540	537
Posible	159	207
Fut_prog	21	8
Total	875	875

Tabla 4 - Valores de factividad de los eventos clasificados por ambas anotadoras

Se observa que la cantidad de eventos anotados como futuro negado y futuro programado son muy pocos.

De los 875 eventos clasificados por ambas anotadoras que contenían valores de factividad 134 difieren en su clasificación. La Tabla 5 muestra la matriz de confusión para estas anotaciones, donde se omiten los eventos con factividad nula.

Real\Predicción	Realizado	No_realizado	Fut_prog	Posible	Fut_neg	Indef
Realizado	501	5	0	24	0	10
No_realizado	1	40	0	0	4	2
Fut_prog	1	0	8	12	0	0
Posible	13	0	0	140	0	6
Fut_neg	0	2	0	0	7	0
Indef	21	1	0	31	1	45

Tabla 5 - Matriz de confusión para los 6 valores de factividad

Dado que **fut_prog**, **posible**, **fut_neg** e **indef** indican todos posibilidad (en lugar de certeza) y dado que la suma total de los eventos clasificados con alguno de estos valores es de 334 o 315, dependiendo de la anotadora, valor bastante inferior a los eventos clasificados como realizado (del orden de los 500) se optó por simplificar el problema y agruparlos todos en una clase llamada **indef**. A su vez se observa que **posible** se confunde bastante con **indef** y muchos de los que son **fut_prog** son clasificados como **posibles**. **Fut_neg** se mezcla tanto con **no_realizado** como con **indef**. Los valores de esta nueva clasificación se muestran en la Tabla 6. Se nota que la cantidad de eventos no realizados es muy inferior a las demás clases.

Factividad	Anotadora 1	Anotadora 2
Indef	288	290
No_realizado	47	48
Realizado	540	537
Total	875	875

Tabla 6 - Cantidad de eventos por valor de factividad clasificados por ambas anotadoras.

De los 875 eventos clasificados por ambas anotadoras que contenían valores de factividad 84 difieren en su clasificación.

Luego de leer los antecedentes notamos que los índices definidos en SIBILA son similares a los marcadores de factividad definidos por Saurí [RS08], por lo tanto podrían ayudar a la hora de determinar la factividad de un evento. Por esto decidimos analizar los índices marcados en el corpus. La Tabla 7 muestra los índices encontrados en el corpus, donde solo una de las anotadoras marcó los índices y lo hizo en solo cuatro archivos. Solo se cuenta con un total de 16 índices distintos, estos son muy pocos como para ayudar a la hora de decidir palabras que marquen la factividad de los eventos.

Clase	Índices	Total
Polaridad	no, nada	2
Eventividad	durante	1
Condición	si, de	2
Modalidad	podía, posibilidad, tiene que, podría, poder, puede, podrá, pueden, hay que, debe, probablemente	11

Tabla 7 - Índices anotados en el corpus

Potenciales inconvenientes

El potencial inconveniente que se observa mirando las estadísticas del tamaño del corpus es que el mismo es bastante reducido comparando con otros corpus que se han utilizado en otras tareas de aprendizaje en PLN. Por ejemplo el trabajo [GSB12] contaba con un corpus de 4596 eventos, es decir, contenía cuatro veces nuestra cantidad de eventos. Otro ejemplo es el corpus de *TimeBank* [TIMEBANK] que tiene 7935 eventos, y aún se considera pequeño (tiene siete veces nuestra cantidad de eventos).

Este problema hace que, al contar con pocos datos, no se cuente con suficiente información representativa como para utilizar en un sistema de aprendizaje. Y esto se da aún más en los eventos clasificados como no realizados, que son muy pocos en comparación con el resto.

5.3. Línea Tope y Línea Base

Para determinar qué tan bien aprende el sistema, es necesario determinar una cota inferior y otra superior, donde un sistema que esté por debajo de la cota inferior no se considera aceptable, y se busca un sistema que esté lo más próximo a la cota superior. Dichas cotas son llamadas línea base y línea tope respectivamente.

5.3.1. Línea Tope

En sistemas en el contexto de PLN no se puede esperar que este asigne siempre el valor correcto a las etiquetas. En particular, dada la ambigüedad del lenguaje, no es realista esperar que dicho sistema sea mejor que el ser humano.

Por esto se decidió determinar la línea tope observando el grado de concordancia que las anotadoras tenían sobre el atributo factividad en los textos del corpus. Para calcular dicha concordancia se consideran solo los archivos del corpus que fueron anotados de manera independiente por las anotadoras. Luego se toma una anotadora como referencia y la otra como si fuera el sistema de etiquetado. Se tomó como referencia a la anotadora 1 y la anotadora 2 como el sistema de etiquetado, esta elección fue realizada de forma arbitraria. Esto es lo mismo que se realizó en [AD10] y en el corpus de TimeBank[4].

Esta concordancia se determinó sólo sobre los eventos que ambas anotadoras habían identificado, descartando aquellos eventos identificados por una pero no la otra.

Luego de obtener los datos del corpus se obtuvo una matriz de confusión de la factividad de los eventos anotados (aquellos anotados por ambas y a las que ambas le habían asignado un valor de factividad) mostrada en la Tabla 8.

Real\Predicción	**Realizado**	**No_realizado**	**Indef**
Realizado	501	5	34
No_realizado	1	40	6
Indef	35	3	250

Tabla 8 - Matriz de confusión para la Línea Tope

Luego se calcularon la precisión, recuperación y Medida F, dichos valores se muestran en la Tabla 9.

[4] http://timeml.org/site/timebank/documentation-1.2.html

	Precisión	Recuperación	Medida F
Realizado	93,3%	92,8%	93%
No_realizado	83,3%	85,1%	84,2%
Indef	86,2%	86,8%	86,5%

Tabla 9 - Precisión, Recuperación y Medida F de la línea Tope

La exactitud de la línea tope es de 90,4%. Cabe destacar que este valor refleja lo compleja que es la tarea de determinar la factividad de los eventos incluso para los seres humanos.

5.3.2. Línea Base

La línea base debe ser simple pero eficiente, es decir que el sistema tenga un mínimo aceptable. Para esto se optó por realizar un algoritmo simple basado en reglas, no basado en aprendizaje automático, cuyo desempeño sea el mínimamente aceptable.

Para la construcción de la línea base se utilizaron dos listas básicas de palabras, una de palabras que indican negación (no_realizado), y otra de palabras que indican posibilidad (indef) basadas en la lista de índices (agregando algunas palabras a las que indican negación: *nunca* y *sin*) extraídas del corpus. Como ya se dijo, se restringe el trabajo a eventos verbales.

Luego de construida la línea base la misma será comparada con la clasificación de una de las anotadoras considerada como "clasificación real" para obtener las métricas de desempeño del mismo. En nuestro caso, dado que una de las anotadoras tenía más eventos clasificados que la otra, y dado que el número de eventos es muy chico, se determinó que dicha anotadora sea tomada como base (anotadora 1).

Primera aproximación a la línea base

En una primera instancia se consideró el árbol de dependencias de cada oración, donde la factividad de un evento E es definida de la siguiente forma:

1. Si alguno de los hijos del árbol de dependencias de raíz E contiene alguno de los elementos de la lista de palabras que indican posibilidad, E es etiquetado como **indef**.

2. Si esto no se cumple, pero si se cumple que alguno de los hijos del árbol de dependencias de raíz E contiene algún elemento de la lista de palabras que indican negación, E es etiquetado como **no_realizado**.

3. Si no se cumple ninguno de estos E es etiquetado como **realizado**.

Se muestra esta estructura en la Figura 1.

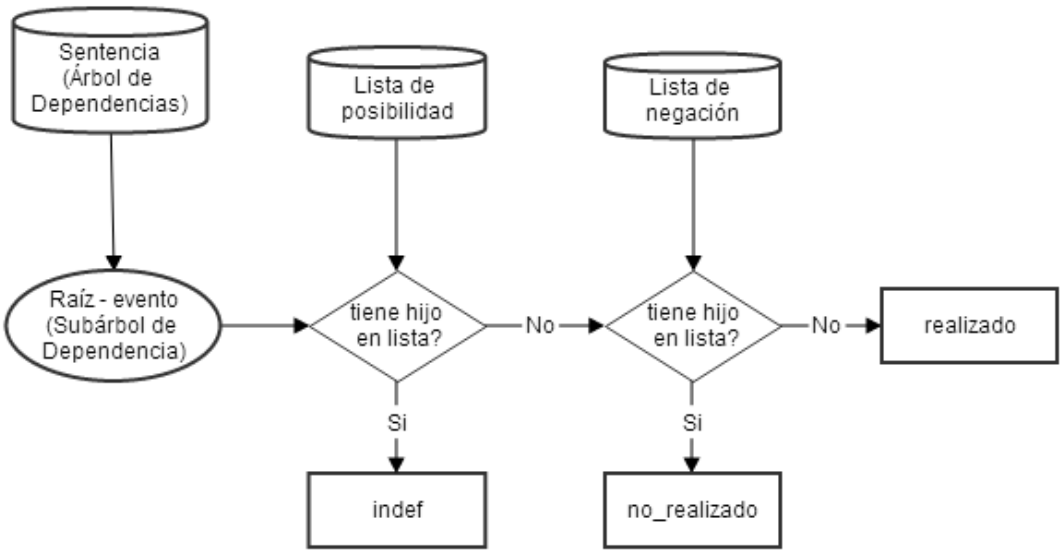

Figura 1 - Diagrama de la línea base inicial

La matriz de confusión de esta primera aproximación de la línea base es mostrada en la Tabla 10, y sus medidas de precisión, recuperación y Medida F en la Tabla 11. De los 800 eventos encontrados, 259 son clasificados erróneamente por la línea base.

Real\Predicción	Realizado	No_realizado	Indef
Realizado	523	25	216
No_realizado	10	17	6
Indef	1	1	1

Tabla 10 - Matriz de confusión de la primera línea base

	Precisión	Recuperación	Medida F
Realizado	97,9%	68,4%	80,6%
No_realizado	39,5%	51,5%	44,7%
Indef	0,44%	33,3%	0,88%

Tabla 11 - Precisión, Recuperación y Medida F de la primera línea base

Dado que esta línea base no produjo resultados aceptables (sobre todo para la factividad **indef**) se decidió implementar una nueva línea base.

Línea Base

La línea base consiste en encontrar los eventos, y determinar su factividad en base al siguiente algoritmo:

1. Si el evento está en tiempo futuro es etiquetado como **indef**.

2. Si esto no se cumple, pero existe alguna palabra que indique negación (pertenece a la lista de palabras que indican negación) en las palabras que estén dentro de la misma oración antes del evento, es etiquetado como **no_realizado**.

3. Si no se cumple ninguno de los anteriores, el evento se etiqueta como **realizado**.

Este algoritmo se muestra en la Figura 2.

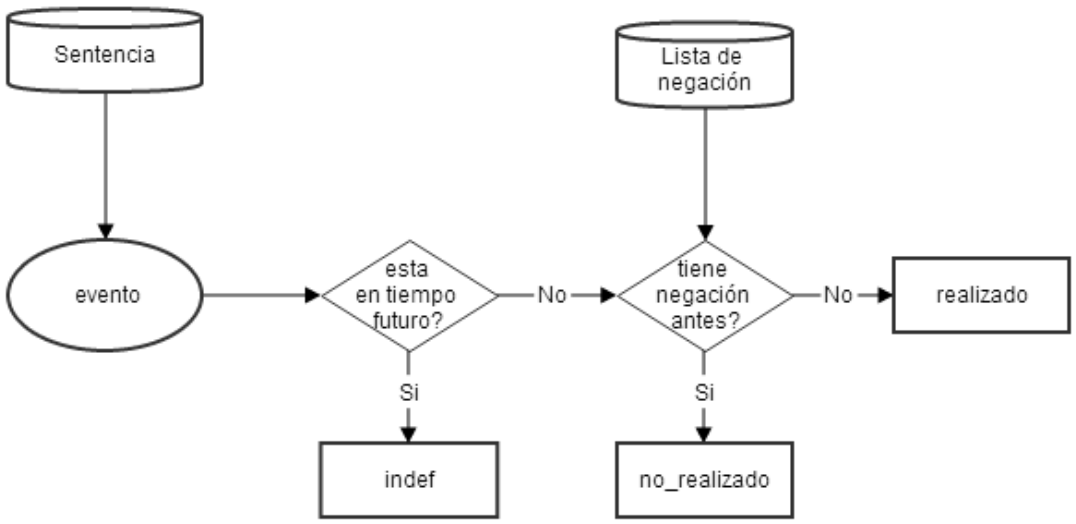

Figura 2 - Diagrama de la línea base

La matriz de confusión de la línea base se muestra en la Tabla 12 y las medidas de precisión, recuperación y Medida F en la Tabla 13. De los 800 eventos encontrados, 252 son clasificados erróneamente.

Real\Predicción	Realizado	No_realizado	Indef
Realizado	469	9	144
No_realizado	65	34	34
Indef	0	0	45

Tabla 12 - Matriz de confusión de la línea base

	Precisión	Recuperación	Medida F
Realizado	87,8%	75,4%	81,1%
No_realizado	79,1%	25,6%	38,6%
Indef	20,2%	100%	33,6%

Tabla 13 - Precisión, Recuperación y Medida F de la línea base

La exactitud de la línea base es de 68,5%.

5.4. Corpus Extendido

Dado que contamos con un corpus muy pequeño decidimos extenderlo de manera de tener más datos de entrenamiento. Para esto se utilizó una versión inicial del sistema entrenado con el corpus original para anotar un conjunto nuevo de artículos de prensa extraídos de los distintos sitios en línea de noticias uruguayas (El País, El Observador, Últimas Noticias, etc.). Gracias a esto se obtuvo un corpus nuevo anotado con una simplificación de la anotación SIBILA orientado a nuestra solución, donde solo se anotaron los eventos (verbos) con su respectiva factividad. Luego se dividió dicho corpus en dos conjuntos de textos, cada uno de los cuales fue corregido manualmente por una de las tutoras de este proyecto.

La Tabla 14 muestra los datos obtenidos de este nuevo corpus, llamado corpus 2. El total de oraciones, eventos y *tokens* de los corpus juntos se muestran en la misma tabla.

	Cantidad de eventos	Cantidad de oraciones	Cantidad de *tokens*
Corpus 2	1274	350	9969
Total (c1+c2)	2341	573	16930

Tabla 14 - Datos del Corpus extendido y datos totales

	Realizado	No_realizado	Indef	Total
Corpus 2	857	76	341	1274
Total (c1+c2)	1392	121	567	2080

Tabla 15 - Factividades del corpus extendido

La Tabla 15, por su lado, muestra las factividades encontradas en el corpus extendido (corpus 2). Se observa que la cantidad de eventos con factividad no_realizado es muy inferior a las otras dos clases. El total de las factividades en ambos corpus también se muestra en la Tabla 15, donde se toma el corpus 1 como el de la anotadora 1, dado que este es el que tiene más *tokens* anotados. Se observa que la cantidad de eventos realizados supera por más del doble a los indefinidos y por más de diez veces a los no realizados.

Si bien consideramos positivo haber aumentado aproximadamente 5/2 del corpus tanto en cada una de las factividades como en el total de eventos, cabe destacar que se sigue teniendo un corpus pequeño para el aprendizaje automático si lo comparamos con el *TimeBank* [TIMEBANK] o el

utilizado en [GSB12]. A su vez se observa que la cantidad de eventos no_realizado sigue siendo muy poca.

Capítulo 6

Solución Propuesta

Se propone construir una herramienta que determine la factividad de los eventos mencionados en un texto utilizando métodos de aprendizaje automático. Para esto se determinó el uso de CRF, SVM y Árboles de decisión como métodos de aprendizaje automático. CRF y SVM fueron elegidos dado que demostraron tener un buen desempeño en sistemas similares y son los más utilizados en el área de PLN. A su vez se probó utilizar árboles de decisión para probar el desempeño de un método poco común en este tipo de problemas, pero fue descartado rápidamente debido a su mal desempeño.

Un problema importante en el aprendizaje automático es la selección de atributos, estos fueron elegidos basados en los trabajos leídos, fueron obtenidos de forma automática gracias a algunas herramientas que se comentan en el siguiente capítulo y fueron luego ajustados según el desempeño obtenido en los experimentos realizados (en el capítulo 8).

6.1. Determinación de los atributos

En las técnicas de aprendizaje automático la elección de atributos es sumamente importante ya que tiene una enorme influencia al momento de obtener un mejor modelo como resultado del entrenamiento. También es importante considerar el tipo de codificación que el atributo debe tener, sea por ejemplo valor binario, categorías, o incluso tokens.

La tarea de seleccionar y descartar atributos es muy importante siendo un problema central en el aprendizaje automático. Este problema es más conocido como *feature selection*.

Los objetivos de la selección de atributos son [Guy03]:

1. Mejorar la predicción de los clasificadores, identificar y tratar los atributos que tienen un mayor peso en la salida del clasificador.

2. Clasificadores más rápidos y más rentables, es decir, que tengan menor costo en cuanto a recursos computacionales, esto se logra al tener que manejar una menor cantidad de atributos.

3. Proporcionar una mejor comprensión del proceso subyacente que se genera con los datos de entrenamiento.

Para la selección de atributos se utilizó un método empírico, resultado de generar una y otra vez casos de prueba y ajuste ante errores, buscando siempre el valor más conveniente para el problema planteado.

En principio se toman como buenos todos los atributos para posteriormente ir descartando los que no aportan al aprendizaje.

En la selección de atributos es importante tener presente un problema común en los algoritmos de aprendizaje conocido como sobreajuste (*overfitting*). El sobreajuste es el efecto de sobreentrenar un algoritmo de aprendizaje, y ocurre cuando se utilizan demasiados atributos, o cuando los datos de entrenamiento son reducidos. Este problema trae como consecuencia que el algoritmo infiera un resultado muy eficiente para los datos del corpus, pero poco eficiente para casos desconocidos.

Se decidió obtener una lista de posibles atributos e ir probándolos para determinar cuáles mejoraban el desempeño del sistema y cuáles no.

6.1.1. Básicos

En primera instancia se buscó trabajar con un conjunto básico de atributos, estos son los atributos más comunes utilizados en PLN.

1. **Id del *token* dentro de la oración.**
   Este atributo, junto con el atributo 6, es necesario para armar el árbol de dependencias. No parece aportar mucho como atributo en sí pero se decidió probar de igual forma por si los clasificadores eran capaces de inferir alguna información sobre la dependencia que hay entre los atributos.

2. ***Token***
   Es el atributo que identifica la palabra en la oración, ya que dos palabras tienen que ser exactamente iguales para tener el mismo *token*.

3. **Lema**
   El lema representa la forma canónica de la palabra. Por esto, este atributo identifica la palabra de forma más general, dado que dos palabras distintas con la misma forma canónica tendrán el mismo lema. Por ejemplo, las palabras *dijo*, *dicen* y *dirán* tienen el mismo lema *decir*.

4. **CPOSTAG**
   El CPOSTAG es la primera columna del POSTAG, que representa la categoría gramatical. Esta parece fundamental a la hora de determinar que un *token* es verbo, lo cual indicaría que el *token* es un evento (según nuestra restricción).

5. **POSTAG**
   El atributo POSTAG determina la información morfológica de la palabra. Esta es interesante ya que determina el tiempo, modo, persona, género, tipo y número de un verbo (que es el caso que nos interesa).

6. **Id del padre en el árbol de dependencias (o 0 si es la raíz).**
   Para este atributo pasa lo mismo que en el atributo 1, solo que además, que el Id del padre sea 0 (porque es la raíz del árbol de dependencias) podría aportar algún tipo de información a los clasificadores.

7. **Relación con el padre en el árbol de dependencias (o ROOT si es raíz).**
   Este atributo representa la relación que tiene el *token* con su padre. Parece interesante, a la hora de determinar la factividad, la relación que existe entre dos verbos, o entre un verbo y sus hijos o padres.

6.1.2. Morfología del Verbo

La elección de trabajar con la morfología del verbo se debe básicamente a que la mayoría de los eventos son verbos con lo cual esta categoría morfológica debería destacar con respecto al resto, además de todo lo mencionado en la sección 4.2.

Por esto se agregan los siguientes atributos:

8. **Si el *token* es evento según nuestra restricción de evento.**
   Este atributo tendrá valor 1 si el token es evento según nuestra restricción, en caso contrario tendrá valor 0. Luego de analizar el corpus se determinó, como ya se dijo, que la mayoría de los eventos eran verbos, es por esto que nuestra primera restricción es que solo marcaremos como eventos a los verbos. También se observó que el corpus contenía una serie de verbos que no eran eventos, estos incluyen los verbos *poder*, *deber*, *tener que* y *haber que* seguidos de un verbo en infinitivo. Es por esto que se decidió que estos no serían considerados eventos, indicando este atributo en 0 para esos casos.

Entrando más en detalle en el trabajo [WMR09], ocurre muy a grandes rasgos que los verbos que se encuentran en tiempo futuro[5] y modo indicativo suelen tomar el valor de indefinido o no realizado. En cambio para los que se encuentran en tiempo pasado y modo indicativo, siempre que no esté en el ámbito de una negación, suelen tomar el valor de realizado. Finalmente, para los que se encuentran en tiempo presente y modo indicativo, siempre que no estén en el ámbito de una negación, suelen tomar el valor de realizado o indefinido. Por todo esto surgió la idea de darle más importancia a este hecho, con lo que se separaron los tiempos en distintos atributos.

Los valores de estos atributos se tomaron de la etiqueta asignada por el analizador morfosintáctico (atributo 4). En el capítulo siguiente se describen las etiquetas que utiliza el analizador que se aplicó.

9. **En caso de ser verbo, atributo que indica si está en tiempo futuro.**
   Si el *token* es un verbo en tiempo futuro tomará el valor 1, si no es verbo o no está en tiempo futuro tomará el valor 0. Este atributo es importante ya que la mayoría de los verbos en tiempo futuro indican que el evento no sucedió todavía (indefinido).

10. **En caso de ser verbo, atributo que indica si está en tiempo pasado.**
    Si el *token* es un verbo en pasado tendrá el valor 1, si no es un verbo o no está en pasado tendrá el valor 0. Un verbo en pasado suele indicar que el evento es realizado o no realizado (si contiene alguna partícula de polaridad negativa), pero generalmente indica certeza (no indefinido).

11. **En caso de ser verbo, atributo que indica si está en tiempo presente.**
    Si el *token* es un verbo en presente tendrá el valor 1, si no es un verbo o no está en presente tendrá el valor 0. Si bien un verbo en presente puede tomar valores de factividad muy variados, decidimos dejar este atributo para ver si influía de alguna forma en nuestro clasificador.

[5] Una excepción a esto es el futuro histórico.

12. **En caso de ser verbo, atributo que indica el modo del verbo.**
    Los posibles valores para este atributo son: I (indicativo), S (subjuntivo), M (imperativo), N (infinitivo), G (gerundio), P (participio), si el token es un verbo, o 0 si no es un verbo. Basados en [WMR09] consideramos el modo como un atributo fundamental a la hora de determinar la factividad de un evento.

13. **En caso de ser verbo, atributo que indica el tiempo del verbo.**
    Los posibles valores para este atributo son: P (presente), I (imperfecto), F (futuro), S (pasado), C (condicional) si el token es un verbo, o 0 si el token no es un verbo. Nuevamente basándonos en [WMR09] parece razonable tener en cuenta el tiempo en el cual nos encontramos. Si bien ya consideramos atributos para el tiempo presente, futuro y pasado, se decidió incluir uno que abarque todos los tiempos.

14. **En caso de ser verbo, atributo que indica si el verbo es condicional.**
    Este atributo tendrá el valor 1 si el token es un verbo en tiempo condicional, o 0 si no es un verbo o no está en condicional. Se agregó también como atributo el verbo en tiempo condicional ya que por lo general, como se muestra en (6.1), este tiempo verbal implica posibilidad y no certeza.

15. **En caso de ser verbo, atributo que indica si el verbo tiene modo indicativo.**
    Este atributo tendrá el valor 1 si el *token* es un verbo en modo indicativo, o 0 si no es un verbo o no es indicativo. Continuando con el razonamiento anterior y viendo la influencia del modo indicativo se agregó también el verbo en modo indicativo.

(6.1) ***Si*** *yo fuera rico, conduciría un deportivo.*

6.1.3. Recursos Léxicos

Un evento realizado se puede ver afectado por una palabra que indica negación convirtiéndolo en no realizado (6.2). De forma similar una palabra que indica posibilidad puede convertir un evento realizado (6.3) o no realizado (6.4) en indefinido. Por todo esto se decidió agregar atributos que faciliten la tarea de nuestros clasificadores al momento de inferir dicha información.

(6.2) *Juan* ***no*** *fue ayer a trabajar.*
(6.3) *Es* ***posible*** *que llueva mañana.*
(6.4) *Es* ***posible*** *que no llueva mañana.*

16. **Indica si el *token* pertenece a la lista de palabras que indican negación.**
    Este atributo toma el valor 1 si el *token* pertenece a la lista de palabras que indican negación y 0 en caso contrario.

17. **Indica si el *token* pertenece a la lista de palabras que indican posibilidad.**
    Este atributo toma el valor 1 si el *token* pertenece a la lista de palabras que indican posibilidad y 0 en caso contrario.

Para estos dos atributos se obtuvo una lista de palabras que indican negación y posibilidad. Esta lista se realizó teniendo en cuenta las palabras mencionadas en [RS08], las mismas se muestran en el anexo en la sección B.

La información de que un *token* pertenece a la lista de negación o posibilidad es relevante cuando miramos el contexto izquierdo y derecho del evento, donde contar con una negación o posibilidad en dichos contextos puede indicar que estamos ante un evento no realizado o indefinido respectivamente.

6.1.4. Subordinados de indicadores de negación y posibilidad

Estos atributos surgen de analizar el contexto que rodea a un evento, se consideraron los siguientes atributos:

18. **Si es un verbo, indica si el árbol de dependencias con raíz el evento tiene un hijo directo que pertenece a la lista de palabras que indican negación.**
    Este atributo toma el valor 1 si el *token* es un verbo y el árbol de dependencias con raíz el verbo tiene un hijo directo que pertenece a la lista de palabras que indican negación, si esto no se cumple, o no es un verbo toma el valor 0. Que un *token* que indique negación se encuentre como hijo del árbol de dependencias con raíz el evento puede indicar que el evento es no realizado. Esto se muestra en la Figura 3 donde se observa el árbol de dependencias de (6.5), en donde la negación *jamás* niega el evento (del cual es hijo) *nieva*.

19. **Si es un verbo, indica si el árbol de dependencias con raíz el evento tiene un hijo directo que pertenece a la lista de palabras que indican posibilidad.**
    Este atributo toma el valor 1 si el *token* es un verbo y el árbol de dependencias con raíz el verbo tiene un hijo directo que pertenece a la lista de palabras que indican posibilidad, si esto no se cumple, o no es un verbo toma el valor 0. De la misma forma, contar con un *token* que indica posibilidad como hijo del subárbol de dependencias de raíz el evento parece indicar que dicho evento se presenta como indefinido. Tal es el caso del ejemplo (6.6) y la Figura 4 que muestra su árbol de dependencias, donde el indicador de posibilidad *posiblemente* indica posibilidad sobre el evento (del que es hijo) *llueva*.

(6.5) *En Uruguay* ***jamás*** *nieva en verano.*
(6.6) ***Posiblemente*** *llueva mañana.*

Nuevamente para estos atributos se utilizaron las listas mencionadas en los atributos anteriores. A su vez se utilizó un árbol de dependencias determinando cuáles eran los hijos directos de cada evento y observando si alguno de ellos pertenecía a cada lista.

Figura 3 - Árbol de dependencias de (6.5)

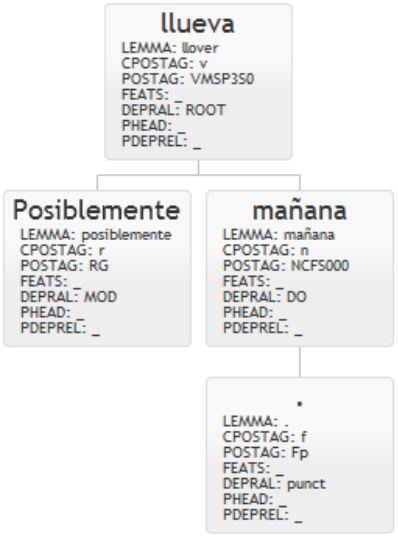

Figura 4 - Árbol de dependencias de (6.6)

6.1.5. Dependencia de indicadores de posibilidad y negación.

De forma similar estos atributos surgen de analizar el contexto que rodea a un evento, obteniendo como resultado los siguientes atributos:

20. **Si es un verbo, indica si en el árbol de dependencias el evento tiene padre o abuelo que pertenece a la lista de negaciones.**
    Toma el valor 1 si el *token* es un verbo y en el árbol de dependencias el verbo tiene padre o abuelo que pertenece a la lista de negaciones, si esto no se cumple o el *token* no es verbo toma el valor 0. Una negación que se encuentra como padre o abuelo del nodo evento dentro del árbol de dependencias puede indicar negación sobre el evento referido. Esto se observa en los ejemplos (6.7) y (6.8) cuyos árboles de dependencias se muestran en las Figuras 5 y 6 respectivamente. En (6.7) el indicador de polaridad *negó* se encuentra como padre del evento *copiado*, indicando la negación de dicho evento. A su vez en (6.8) el indicador de polaridad *evitó*, que es el abuelo del evento *explotara*, indica negación sobre dicho evento.

21. **Si es un verbo, indica si en el árbol de dependencias el evento tiene padre o abuelo que pertenece a la lista de posibilidades.**
    Toma el valor 1 si el *token* es un verbo y en el árbol de dependencias el verbo tiene padre o abuelo que pertenece a la lista de posibilidades, si esto no se cumple o el *token* no es verbo toma el valor 0. Un caso similar al atributo anterior se da con las listas de posibilidades. Esto se observa en los ejemplos (6.9) y (6.10) junto con las Figuras 7 y 8 que muestran sus árboles de dependencias. En (6.9) por un lado el indicador de posibilidad *procuró* se encuentra como padre del evento *lavar* indicándolo como indefinido. Por otro lado, en (6.10), el indicador de posibilidad *parecer* se encuentra como abuelo del evento *abierta* indicándolo también como indefinido.

22. **Si es un verbo, indica si en el árbol de dependencias el evento tiene algún hijo de antecesor con menor id (atributo 1 del antecesor menor que el atributo 1 del *token* actual) que pertenece a la lista de negaciones.**
    Toma el valor 1 si el *token* es un verbo y en el árbol de dependencias el verbo tiene algún "hermano" que pertenezca a la lista de negaciones, si esto no se cumple o el *token* no es verbo toma el valor 0. Puede suceder que el indicador de negación se encuentre como "hermano" del evento, es decir, como un antecesor que tiene un id (atributo 1) menor. Esto se observa en el ejemplo (6.11) y la Figura 9 que contiene su árbol de dependencias, donde el indicador *nada* se encuentra como "hermano" del evento *reír*, indicándolo como no realizado.

23. **Si es un verbo, indica si en el árbol de dependencias el evento tiene un hijo de un antecesor con menor id (atributo 1 del antecesor menor que el atributo 1 del *token* actual) que pertenece a la lista de posibilidades.**
    Toma el valor 1 si el *token* es un verbo y en el árbol de dependencias el verbo tiene algún "hermano" que pertenezca a la lista de posibilidades, si esto no se cumple o el *token* no es verbo toma el valor 0. De igual forma que el atributo anterior, puede suceder que el indicador de posibilidad se encuentre como "hermano" del evento indicándolo como

indefinido. Tal es el caso del ejemplo (6.12) y la Figura 10 que contiene su árbol de dependencias, donde el indicador *sospecha* se encuentra como "hermano" del evento *robado*.

Estos cuatro atributos son determinados con las mismas listas mencionadas anteriormente pero en lugar de fijarnos en sus hijos nos fijamos en sus antecesores y "hermanos".

(6.7) *Juan* ***negó*** *haber* *copiado* *en el examen.*
(6.8) *Se* ***evitó*** *que* *explotara* *la granada en Iraq.*
(6.9) *María* ***procuró*** *lavar* *los pisos.*
(6.10) *Al* ***parecer*** *la ventana está* *abierta*.
(6.11) ***Nada*** *le hace* *reír*.
(6.12) *El* ***sospecha*** *que fue* *robado*.

Figura 5 - Árbol de dependencias de (6.7)

Figura 6 - Árbol de dependencias de (6.8)

Figura 7 - Árbol de dependencias de (6.9)

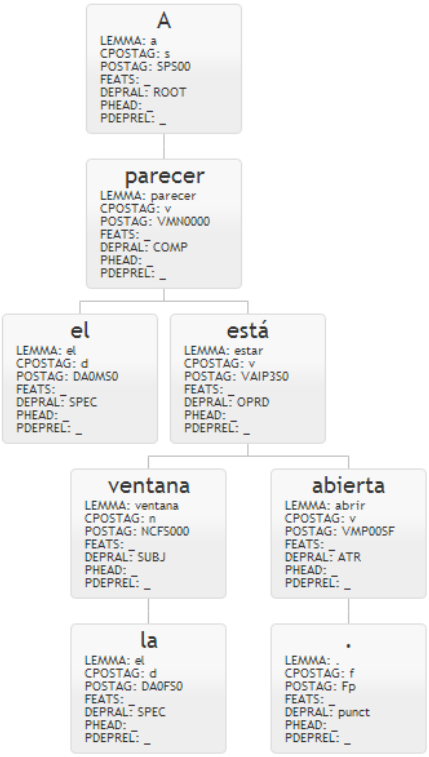

Figura 8 - Árbol de dependencias de (6.10)

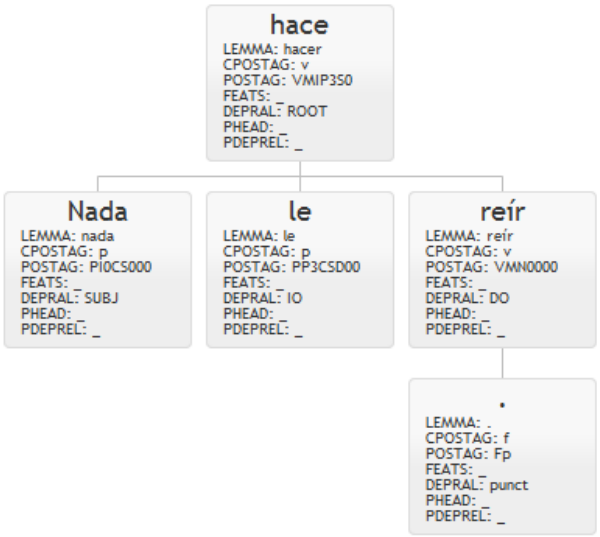

Figura 9 - Árbol de dependencias de (6.11)

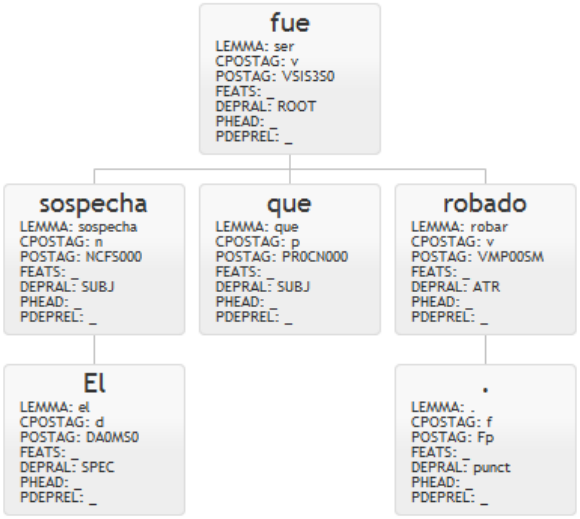

Figura 10 - Árbol de dependencias de (6.12)

6.1.6. Predicados implicativos

A partir del análisis del trabajo de [MW13], el cual propone que la implicación es una propiedad lógica que tienen algunos verbos de afectar la veracidad y/o falsedad de otro verbo que depende de ellos, se plantea la existencia de cuatro casos que nos resultaron interesantes:

- IP++: Este tipo de predicado trae como implicancia que el evento que depende de él tiene polaridad positiva si la polaridad del contexto de dicho predicado es positiva. Un ejemplo de esto se muestra en (6.13) donde el predicado *comprobó*, que tiene polaridad positiva, afecta al evento *abierta* indicando que dicho evento tiene polaridad positiva (realizado). Es decir, la puerta estaba abierta.

- IP+-: Este tipo de predicado trae como implicancia que el evento que depende de él tiene polaridad negativa si la polaridad del contexto de dicho predicado es positiva. Un ejemplo de esto se muestra en (6.14) donde el predicado implicativo *descartó*, que tiene polaridad positiva, afecta al evento *abierta* indicando su polaridad como negativa (no realizado). Es decir, la puerta no estaba abierta.

- IP-+: Este tipo de predicado trae como implicancia que el evento que depende de él tiene polaridad positiva si la polaridad del contexto de dicho predicado es negativa. Un ejemplo de esto se muestra en (6.15) donde el predicado implicativo *imaginó*, que tiene polaridad negativa gracias al indicador *nunca*, afecta al evento *abierta* indicando su polaridad como positiva (realizado). Es decir, la puerta estaba abierta.

- IP--: Este tipo de predicado trae como implicancia que el evento que depende de él tiene polaridad negativa si la polaridad del contexto de dicho predicado es negativa.

Un ejemplo de esta implicación se muestra en (6.16) donde el predicado *intentó*, que tiene polaridad negativa gracias al indicador *no*, afecta al evento *abrir* indicando su polaridad como negativa (no realizado). Es decir, juan no abrió la puerta.

(6.13) *José **comprobó** que la puerta estaba abierta.*
(6.14) *José **descartó** que la puerta estuviera abierta.*
(6.15) *José nunca **imaginó** que la puerta estaba abierta.*
(6.16) *Juan no **intentó** abrir la puerta.*

Luego de observar estos casos surgió la idea de considerar cuatro listas extraídas de [MW13], y agregar cuatro atributos que indican si el verbo pertenece a alguna de las listas de implicaciones IP++, IP+-, IP-+ o IP--, estas listas se encuentran en el anexo en la sección B.

A continuación se detallan dichos atributos, estos simplemente indican si los *tokens* pertenecen o no a cada uno de las listas de implicaciones. Nuevamente indicando que un *token* pertenece a una lista de implicación resulta útil al contemplar el contexto izquierdo y derecho del evento.

24. **Si es un verbo, indica si el *token* pertenece a la lista de implicaciones + + (IP++).**

25. **Si es un verbo, indica si el *token* pertenece a la lista de implicaciones + - (IP+-).**

26. **Si es un verbo, indica si el *token* pertenece a la lista de implicaciones - + (IP-+).**

27. **Si es un verbo, indica si el *token* pertenece a la lista de implicaciones - - (IP--).**

6.1.7. Atributo implicación

Por último, dado que la implicación puede darse de forma encadenada, se agregó un último atributo, el cual determina la polaridad del verbo dependiendo de la implicación y polaridad de sus antecesores.

28. **Si es un verbo, determinamos la polaridad del mismo dependiendo de las implicaciones de sus antecesores. Su valor es POS, si se determina por implicancias que su polaridad es positiva, NEG si es negativa y 0 si no se puede determinar su polaridad o no es un verbo.**
    Este atributo también indica si los *tokens* pertenecen o no a cada una de las listas de implicaciones mencionadas anteriormente y se basa en las reglas de implicaciones. Para esto debe fijarse si su padre o abuelo pertenece a alguna de estas listas de implicancias y dependiendo de la implicancia y su polaridad, la polaridad del evento. Se observa que esta regla es recursiva, ya que determinar la polaridad de un padre implica determinar a su vez si este tiene un padre o abuelo que pertenece a alguna de las listas de implicancias. Se explica cómo se calcula este atributo en la Figura 11, donde no se muestra la recursividad, pero esta ocurre al calcular la polaridad del antecesor. En general, la polaridad de un antecesor se calcula:

    1- Si tiene un antecesor que pertenece a la lista de implicaciones: se calcula la polaridad dependiendo de estas implicaciones.

    2- En caso contrario, si esta negado, su polaridad es negativa.

3- En caso contrario, es positiva.

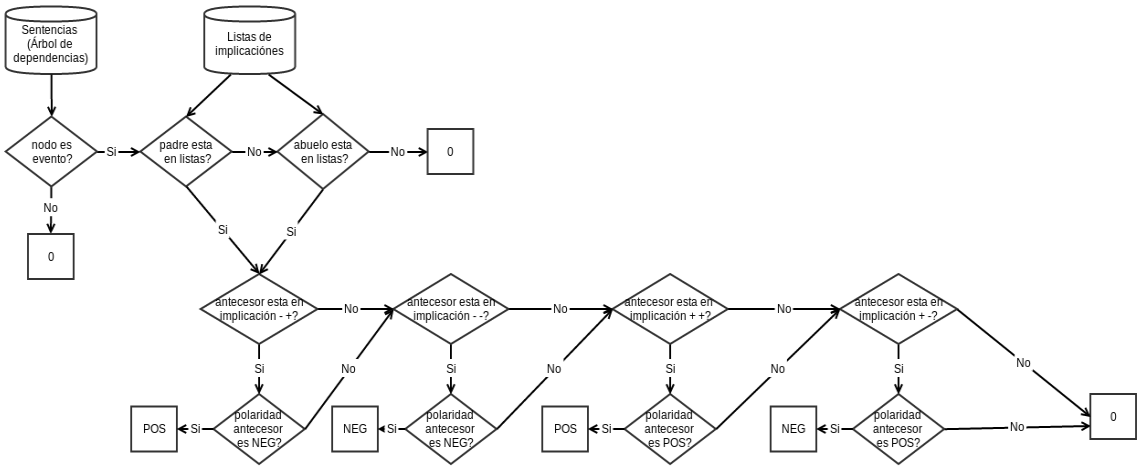

Figura 11 - Atributo 28

Se consideraron los padres y abuelos de los eventos porque se encontraron ejemplos para ambos casos. Un ejemplo de un verbo implicativo que es padre del evento es (6.17) cuyo árbol de dependencias se muestra en la Figura 12, donde el padre del evento *abrir* es un predicado de implicación - - (*lograr*) por lo que al estar negado se induce que la puerta no se abrió. Por otro lado, un ejemplo de un verbo implicativo que es abuelo del evento es (6.18), cuyo árbol de dependencias se muestra en la Figura 13, donde el predicado *olvidar* (que tiene polaridad positiva) tiene implicación + - por lo que el evento *cerrar la puerta* tiene polaridad negativa (Juan no cerró la puerta).

(6.17) *Juan no **logró** abrir la puerta.*
(6.18) *Juan se **olvidó** de cerrar la puerta.*

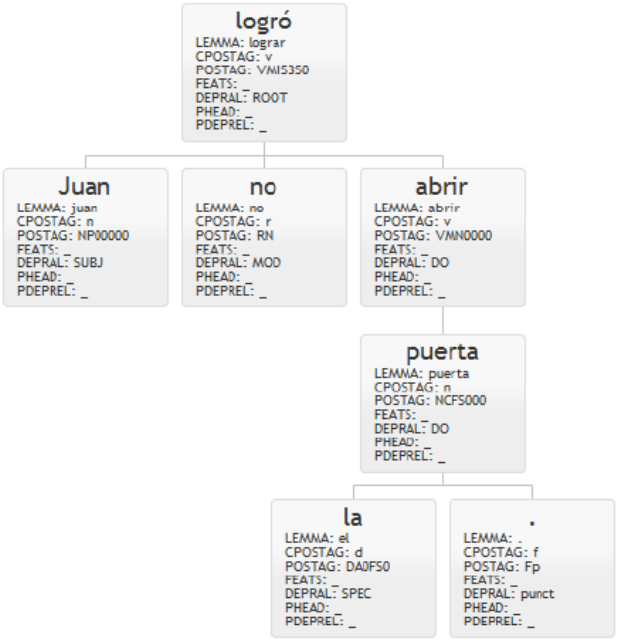

Figura 12 - Árbol de dependencias de (6.17)

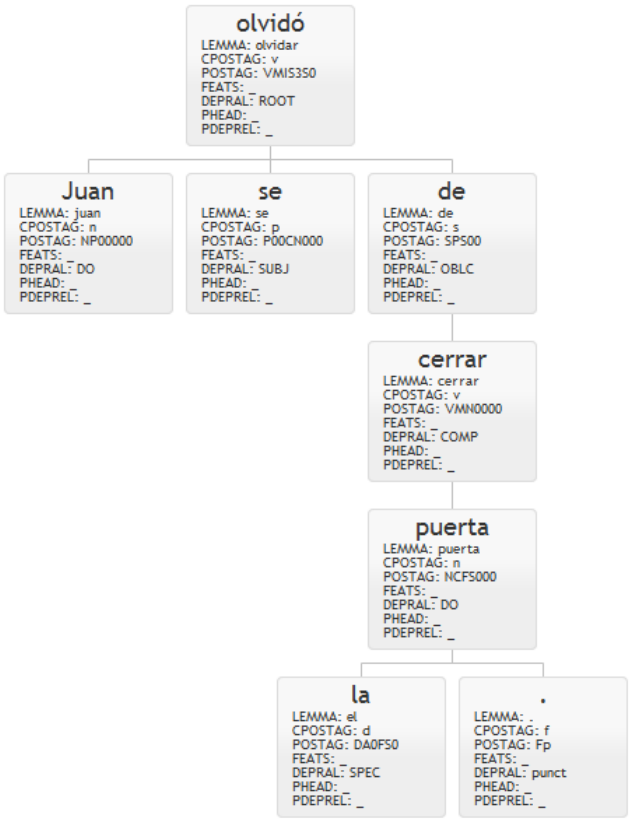

Figura 13 - Árbol de dependencias de (6.18)

6.2. Clasificadores

Para la elección de los clasificadores se buscó probar diversos métodos de aprendizaje automático. Se optó por utilizar *Conditional Random Fields* (CRF) y *Support Vector Machine* (SVM) ya que en investigaciones similares a la factividad de un evento o afines al área de PLN se han obtenido resultados satisfactorios. También se optó por probar con *Decision Trees* (DT) con el fin de experimentar con un método totalmente diferente y ampliar el espectro de soluciones.

6.2.1. *Conditional Random Fields* (CRF)

Es un método de modelado estadístico aplicado comúnmente en sistemas de reconocimiento de patrones y métodos de aprendizaje automático.

Mientras que un clasificador común predice la etiqueta de una sola muestra sin tener en cuenta a sus "vecinos", un CRF puede considerar el contexto. En el caso de una cadena lineal para el procesamiento del lenguaje natural, CRF predice secuencias de etiquetas a partir de secuencias de muestras de entrada.

[LMCP01] definen un CRF basándose en observaciones X y variables aleatorias Y como:

Sea $G = (V, E)$ un grafo tal que $Y = (Y_v)_{v \epsilon V}$, de tal forma que Y está indexado en los vértices de G. Entonces (X, Y) es un CRF cuando las variables aleatorias Y_v, condicionado en X, obedecen la propiedad de Markov con respecto al grafo: $p(Y_v | X, Y_w, w \neq v) = p(Y_v | X, Y_w, w \sim v)$ donde $w \sim v$ significa que w y v son vecinos en G. Así, un CRF es un campo aleatorio global condicionado a la observación X.

Los CRF representan toda la información mediante funciones características (*feature functions*) [KA11]. La interpretación de los observables X y las definiciones de la estructura del problema (grafo de salida) son modeladas por esta función.

Las funciones características son funciones indicadoras de un suceso, una cualidad o una propiedad, y toman únicamente los valores 0 y 1 (binarios). Estas indican la pertenencia (o no) a un conjunto.

Un CRF por definición impone una única restricción que es la de independencia entre los elementos contenidos en Y que no sean vecinos, o sea que en una función característica podemos utilizar cualquier información:

- Extraída de uno o varios observables (elementos de X).
- Sobre un cierto elemento de Y.
- Sobre dos elementos de Y que sean vecinos entre sí, si y solo si al definir una función que utilice dos elementos de Y estamos indicando la existencia de una arista (dependencia) entre ellos.
- Combinación de los anteriores.

Las funciones características son útiles para introducir conocimiento de experto en el modelo, estas permiten definir nueva información derivada a partir de toda la entrada X, así como nuevas relaciones de interés entre los elementos de la salida Y.

6.2.2. *Support Vector Machine* (SVM)

Los SVM son un grupo de métodos de aprendizaje supervisado que se pueden aplicar a la clasificación o a la regresión. El algoritmo de SVM se basa en la teoría del aprendizaje estadístico y la dimensión de Vapnik-Chervonenkis (VC). Son un conjunto de algoritmos que aprenden de los datos mediante la creación de modelos que maximizan el margen de error de un conjunto de entrenamiento.

Es decir, un SVM es un modelo que representa a los puntos de muestra en el espacio, separando las clases por un espacio lo más amplio posible. Cuando las nuevas muestras se ponen en correspondencia con dicho modelo, en función de su proximidad pueden ser clasificadas a una u otra clase. Una buena separación entre las clases permitirá una clasificación correcta.

Para el caso de SVM lineal, dados un conjunto de entrenamiento D, un conjunto de n puntos de la forma $D = \{ (x_i, y_i) \mid x_i \in R^P, y_i \in \{-1, 1\} \}_{i=1}^{n}$ donde y_i es 1 o -1, indicando la clase a la que x_i pertenece, y cada x_i es un vector real de p dimensiones. La idea de SVM es encontrar el hiperplano de mayor margen que divide los puntos que tienen $y_i = 1$ de los que tienen $y_i = -1$. Cualquier hiperplano puede ser descrito como un conjunto de puntos x que satisfacen $w.x - b = 0$ donde "." denota el producto escalar y w el vector normal al hiperplano.

El parámetro $\frac{b}{||w||}$ determina el desplazamiento del hiperplano desde el origen a lo largo del vector normal w. Si los datos de entrenamiento son linealmente separables se pueden seleccionar dos hiperplanos de manera que separen los datos y que no haya puntos en el medio y luego maximizar su distancia. La región acotada por ellos se llama "el margen". Estos hiperplanos se describen como $w.x - b = 1$ y $w.x - b = -1$.

La distancia entre estos dos hiperplanos es $\frac{2}{||w||}$ por lo que se quiere maximizar $||w||$.

Para prevenir que los puntos de datos caigan en el margen se agregan las siguientes restricciones: para cada i, $w.x_i - b \geq 1$ para las x_is de la primera clase o $w.x_i - b \leq -1$ para las x_is de la segunda. Es decir, $y_i(w.x_i - b) \geq 1$ para todo $1 \leq i \leq n$.

La Figura 14 muestra un ejemplo de esto, donde se muestran los hiperplanos que separan los datos y el hiperplano de mayor margen.

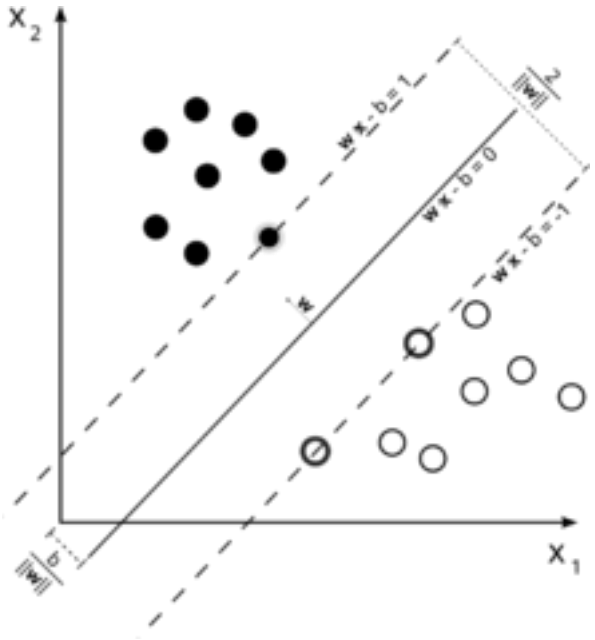

Figura 14 - Ejemplo de SVM

6.2.3. *Decision Tree Learning* (DTL)

En *Decision Tree Learning* se utiliza un árbol de decisión como un modelo predictivo que mapea observaciones acerca de un elemento a conclusiones sobre el valor objetivo de dicho elemento. Es uno de los enfoques de modelado predictivo usado en estadísticas, la minería de datos y aprendizaje automático, además es una de las técnicas de inferencia inductiva más usada.

En la estructura de árbol, las hojas representan las etiquetas de clase y las ramas conjunciones de características que conducen a esas etiquetas de clase. El objetivo es crear un modelo que predice el valor de una variable de destino en función de diversas variables de entrada.

Un árbol se puede "aprender" mediante el fraccionamiento de un conjunto fuente en subconjuntos, basados en un análisis del valor del atributo. Este proceso se repite en cada subconjunto derivado de una manera recursiva llamada particionamiento recursivo. La recursividad es terminada cuando el subconjunto en un nodo tiene todos los mismos valores en las variables objetivos, o cuando la división ya no agrega valor a las predicciones.

Los datos provienen de los registros de la forma: $(x,Y) = (x_1, x_2, x_3, ..., x_k, Y)$

La variable dependiente, Y, es la variable objetivo que estamos tratando de predecir, clasificar y generalizar. El vector x se compone de las variables de entrada, x_1, x_2, x_3, etc., que se utilizan para esa tarea.

Un ejemplo de un árbol de decisión para nuestra línea base sería la Figura 15 donde se observa que para determinar la factividad se considera el caso en que el verbo está en futuro tomando el valor indef y en caso contrario, si tiene alguna negación antes toma el valor no_realizado, y si no tiene negación toma el valor realizado.

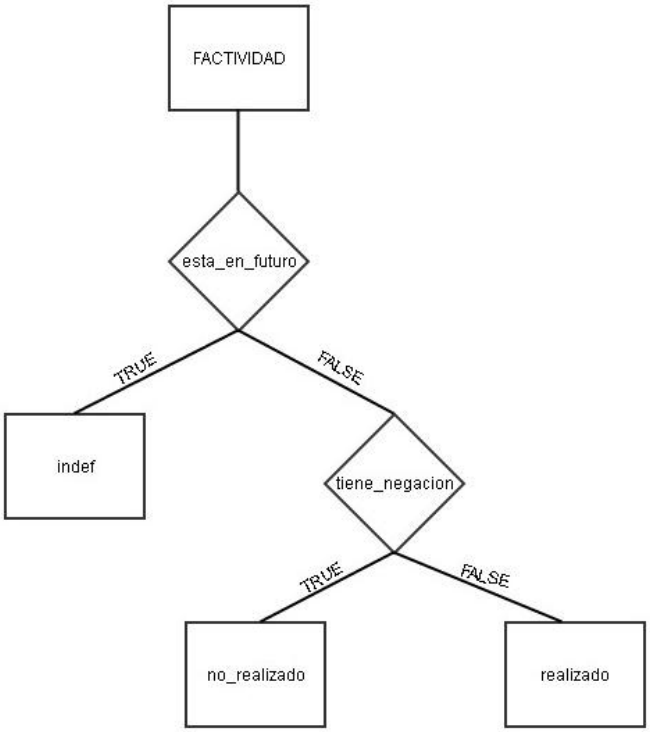

Figura 15 - Ejemplo de un árbol de decisión

6.2.3.1. C4.5

El algoritmo C4.5 se utiliza para generar un árbol de decisión que puede ser utilizado para clasificar, por esto es comúnmente llamado clasificador estadístico.

Este algoritmo toma un conjunto de entrenamiento y construye un árbol de decisión basándose en el concepto de entropía de información. El conjunto de entrenamiento es un grupo $S = \{s_1,s_2,\ldots\}$ de ejemplos ya clasificados que son utilizados para el aprendizaje. Luego cada uno de los ejemplos $s_i = \{x_1,x_2,\ldots\}$ es un vector donde $x_1,x_2,\ldots$ son los atributos del ejemplo. Además el conjunto de entrenamiento es aumentado con un vector $C=\{c_1,c_2,...\}$ donde $c_1,c_2,....$ representa la clase asociada que tiene cada ejemplo del conjunto.

La base del algoritmo es elegir para cada nodo del árbol el atributo del conjunto que mejor divide el conjunto de entrenamiento en subconjuntos, utilizando el criterio normalizado para ganancia de información, la diferencia de entropía. El atributo con mayor ganancia de información es elegido como parámetro de decisión.

Luego seguirá recursivamente en los siguientes subnodos hasta llegar a uno de los pasos bases que se detallan a continuación:

- Todas las muestras en la lista pertenecen a la misma clase. Cuando esto sucede, simplemente crea un nodo de hoja para el árbol de decisión diciendo que elija esa clase.

- Ninguna de las características proporciona ninguna ganancia de información. En este caso, C4.5 crea un nodo de decisión más arriba del árbol utilizando el valor esperado de la clase.

- Instancia de la clase previamente no vista encontrada. Una vez más, C4.5 crea un nodo de decisión más arriba en el árbol con el valor esperado.

Capítulo 7

Implementación

Se presentan en este capítulo las herramientas utilizadas para la implementación de la solución. En particular se explican las librerías Freeling y Maltparser utilizadas para el procesamiento del corpus, y los toolkits CRF++, Yamcha y WEKA. A su vez, también mostramos los *scripts* realizados para la implementación del sistema.

7.1. Conceptos básicos

7.1.1. CoNLL

CoNLL[6] (*Conference on Computational Natural Language Learning*) es una conferencia internacional para la investigación sobre el aprendizaje del lenguaje natural. En la misma se presentan trabajos sobre todos los aspectos de los enfoques computacionales para el aprendizaje del lenguaje natural.

En la CoNLL 2007 [CoNLL07] se propone un formato [DFCoNLL] donde cada oración es representada como una secuencia de *tokens* seguidos de características adicionales como: lema, etiquetado POS, o propiedades morfológicas. Para cada *token*, el parseador debe retornar su padre (dentro del árbol de dependencias) y la correspondiente relación de dependencia. La Tabla 16 muestra las columnas que debe tener cada fila, donde cada fila representa un *token*.

Número de columna	**Nombre de la columna**	**Descripción**
1	ID	Contador de *token*, empieza en 1 para cada oración nueva.
2	FORM	La palabra o símbolo de puntuación.
3	LEMMA	Lema de la forma del *token* o _ si no está disponible.
4	CPOSTAG	Etiqueta POS genérica, generalmente es la primera letra de POSTAG
5	POSTAG	Etiqueta POS particular o CPOSTAG si no está disponible.
6	FEATS	Conjunto desordenado de rasgos sintácticos y/o morfológicos (dependiendo del idioma en particular), separados por una barra vertical (|), o un guion (_) si no está disponible.
7	HEAD	Id del "padre" del *token* en el árbol de dependencias o cero (0) si es la raíz.

[6] http://www.conll.org/

8	DEPREL	Relación de dependencia al padre. El conjunto de las relaciones de dependencia depende del idioma en particular.
9	PHEAD	Cabeza proyectiva del *token* actual, que puede ser un valor de ID o cero (0) o un guion si no está disponible.
10	PDEPREL	Relación de dependencia con el PHEAD o un guion (_) si no está disponible.

Tabla 16 - Formato CoNLL

Este formato es el utilizado en los atributos básicos mencionados en la sección 6.1.1.

7.2. Freeling

Freeling es una herramienta *open source* que provee servicios de análisis del lenguaje, soporta múltiples lenguajes, entre los cuales se encuentran el inglés y el español. Algunos de los servicios que provee Freeling son: analizador morfológico, etiquetado gramatical y analizador de dependencias.

Freeling es una librería sobre la cual se pueden desarrollar aplicaciones de PLN, y está orientado a facilitar la integración con aplicaciones escritas en Perl, Python, Java, Ruby o Php.

7.2.1. Etiquetado POS (*Part Of Speech*)

Es el proceso de asignar (o etiquetar) a cada uno de los *tokens* de un texto su categoría gramatical. Este proceso determina el valor de la etiqueta POSTAG mostrada en la sección anterior.

Dado el ejemplo (7.1) el etiquetador gramatical devolverá la salida mostrada en la Tabla 17.

(7.1) *Estaba lloviendo en Montevideo pero no llovía en Canelones.*

Token	**Lema**	**POSTAG**
Estaba	estar	VAII1S0
lloviendo	llover	VMG0000
en	en	SPS00
Montevideo	montevideo	NP00000
pero	pero	CC
no	no	RN
llovía	llover	VMII3S0
en	en	SPS00
Canelones	canelones	NP00000
.	.	Fp

Tabla 17 - Etiquetado POS de la frase "*Estaba lloviendo en Montevideo pero no llovía en Canelones*."

Esta herramienta es la utilizada para obtener los atributos 2 a 5 mencionados en la sección 6.1.1. A su vez, nos es útil a la hora de determinar los eventos en la línea base, ya que basta con fijarse si el etiquetado POS del *token* empieza con V.

7.2.1.1. Etiquetas EAGLES

La etiqueta POS se basa en las etiquetas propuestas por el grupo [EAGLES] para la anotación de los corpus para todas las lenguas europeas. Las etiquetas EAGLES son utilizadas por Freeling, estas representan la información morfológica de las palabras.

La primera columna de esta etiqueta representa la categoría gramatical, y los posibles valores se muestran en la siguiente tabla.

Categoría	**Valor de la columna**
Adjetivos	A
Adverbios	R
Determinantes	D
Nombres	N
Verbos	V
Pronombres	P
Conjunciones	C
Interjecciones	I
Preposiciones	S
Signos de Puntuación	F
Numerales	Z
Fechas y Horas	W

Tabla 18 - Valor de la primera columna del etiquetado POS asociado a su significado

Esta columna nos es importante a la hora de determinar cuáles son los eventos del texto, donde, al considerar solo los verbos, nos fijamos que esta columna contenga el valor “V”. El resto de las columnas identifican otras características de la palabra como ser: tiempo, modo, persona. Dado que solo nos interesan los verbos se muestra en la siguiente tabla los posibles valores del resto de las columnas de la etiqueta EAGLE solo para los verbos.

Posición	Atributo	Valor	Código
1	Categoría	Verbo	V
2	Tipo	Principal	M
		Auxiliar	A
		Semiauxiliar	S
3	Modo	Indicativo	I
		Subjuntivo	S
		Imperativo	M
		Infinitivo	N
		Gerundio	G
		Participio	P
4	Tiempo	Presente	P
		Imperfecto	I
		Futuro	F
		Pasado	S
		Condicional	C
		-	0
5	Persona	Primera	1
		Segunda	2
		Tercera	3
6	Número	Singular	S
		Plural	P
7	Género	Masculino	M
		Femenino	F

Tabla 19 - Atributos del POSTAG para verbos

7.2.2. Analizador de dependencias

El analizador de dependencias construye un árbol de dependencias del texto seleccionado. Se muestra como queda la salida del analizador de dependencias del mismo ejemplo de la parte anterior (7.1) en la Figura 16.

Figura 16 - Árbol de dependencias del ejemplo (7.1)

Luego de probar este servicio se concluyó que, a pesar de que el desempeño es bastante bueno, tiene una gran falla, y es que el árbol no devuelve el orden del *token* dentro de la oración, que se verá luego que es necesario para poder armar los archivos de entrada de CRF y SVM.

7.3. MaltParser

MaltParser [M1.7.2] es un sistema para generar analizadores de dependencias basado en aprendizaje automático. Dicho aprendizaje se realiza utilizando un corpus donde cada frase fue parseada previamente. Es un algoritmo *greedy* que toma decisiones locales óptimas y es guiado por un clasificador entrenado con secuencias de derivaciones.

Sus principales ventajas son que es fácil de implementar, eficiente en tiempo (determinístico), necesita pocos datos para entrenarlo (su curva de aprendizaje es empinada) y se aprende la historia de las transiciones de un *parser* determinista.

Una gran ventaja que presenta el analizador de dependencias de MaltParser frente al de Freeling es que permite determinar las dependencias de un texto en formato CONLL. Por esto se decidió utilizar el analizador de dependencias de MaltParser en lugar del de Freeling.

Dado el mismo ejemplo mostrado en Freeling (7.1) la salida de MaltParser en formato CONLL sería:

1	Estaba	estar	v	VAII1S0	_	0	ROOT	_	_
2	lloviendo	llover	v	VMG0000	_	1	COMP	_	_
3	en	en	s	SPS00	_	2	MOD	_	_
4	Montevideo	montevideo	n	NP00000	_	3	COMP	_	_
5	pero	pero	c	CC	_	1	COORD	_	_
6	no	no	r	RN	_	7	MOD	_	_
7	llovía	llover	v	VMII3S0	_	5	CONJ	_	_
8	en	en	s	SPS00	_	7	MOD	_	_
9	Candelones	canelones	n	NP00000	_	8	COMP	_	_
10	.	.	f	Fp	_	9	punct	_	_

Se observa que las columnas 2, 3 y 5 son las mismas que la salida del etiquetado POS de Freeling (Tabla 17). Esto se debe a que Maltparser utiliza Freeling para obtener el POSTAG.

Este formato transformado a un árbol de dependencias se muestra en la Figura 17.

Figura 17 - Árbol de dependencias generado por MaltParser para el ejemplo (7.1)

Se observa que los árboles difieren, en general, comparando algunos ejemplos, se encontró que MaltParser presentaba mejores resultados que el analizador de dependencias de Freeling.

Maltparser es la herramienta utilizada para determinar los atributos 6 y 7 mencionados en la sección 6.1.1.

7.4. Toolkits

La tarea de implementar un algoritmo de aprendizaje automático es sumamente compleja por lo cual es necesaria la utilización de *toolkits* que faciliten dicha tarea. Un *toolkit* en la computación es un conjunto de programas con un propósito particular.

Dado que tenemos tres tipos de clasificadores que nos interesa aplicar a nuestro corpus buscamos *toolkits* que nos facilitaran dichas tareas. Para el clasificador de CRF utilizamos el llamado CRF++, para SVM utilizamos Yamcha y para DT utilizamos Weka.

7.4.1. CRF++

CRF++[7] es una implementación *open source* de CRF para etiquetar datos secuenciales. Está diseñado para uso genérico, que puede ser aplicado a una variedad de tareas de PLN.

Tanto los archivos para el entrenamiento como para el testeo, deben consistir de múltiples *tokens*, donde cada *token* consiste de un número fijo de columnas. Cada *token* ocupa una línea y está compuesto, generalmente, de palabras, donde las columnas están separadas por espacios en blanco (espacios o tabuladores). Un conjunto de *tokens* se transforma en una oración, y cada oración debe estar separada por una línea en blanco.

Los archivos pueden tener tantas columnas como se quieran, pero la cantidad de columnas debe ser fija para todos los *tokens*. A su vez, las columnas tienen un mismo significado para todos los *tokens*, por ejemplo la primera columna podría ser siempre el *token*, la segunda la etiqueta de POS y la tercera la subcategoría de POS. La última columna representa la etiqueta que será entrenada por CRF++.

[7] http://crfpp.googlecode.com/svn/trunk/doc/index.html

1	Los	el	d	DA0MP0	O
2	Organismos	organismo	n	NCMP000	O
3	Internacionales	internacional	a	AQ0CP0	O
4	Económicos	económico	a	AQ0MP0	O
5	,	,	f	Fc	O
6	La	el	d	DA0FS0	O
7	UE	ue	n	NP00000	O
8	Y	y	c	CC	O
9	EEUU	eeuu	n	NP00000	O
10	han	haber	v	VAIP3P0	realizado
11	demostrado	demostrar	v	VMP00SM	realizado
12	una	uno	d	DI0FS0	O
13	ceguera	ceguera	n	NCFS000	O
14	ante	ante	s	SPS00	O
15	La	el	d	DA0FS0	O
16	corrupción	corrupción	n	NCFS000	O
17	política	político	a	AQ0FS0	O
18	presente	presente	n	NCMS000	O
19	En	en	s	SPS00	O
20	La	el	d	DA0FS0	O
21	extinta	extinto	a	AQ0FS0	O
22	URSS	urss	n	NP00000	O
23	que	que	p	PR0CN000	O
24	sólo	sólo	r	RG	O
25	siendo	ser	v	VSG0000	indef
26	voluntaria	voluntario	a	AQ0FS0	O
27	resulta	resultar	v	VMIP3S0	indef
28	comprensible	comprensible	a	AQ0CS0	O
29	.	.	f	Fp	O

Figura 18 - Ejemplo de archivo de entrenamiento para CRF++ o Yamcha

Se muestra en la Figura 18 un ejemplo de un posible archivo de datos para CRF++, éste tiene 6 columnas:

1. el identificador del *token* dentro de la oración
2. el *token*
3. lema del *token*
4. CPOSTAG (primera letra del POSTAG)
5. POSTAG
6. etiqueta

7.4.2. Yamcha

Yamcha es un clasificador de texto *open source*, genérico y personalizable, orientado a muchas tareas de PLN, como etiquetado gramatical, reconocimiento de entidades, etc. Yamcha utiliza el algoritmo de aprendizaje de máquinas de estado SVM.

El formato de los archivos de entrenamiento y testeo es el mismo que utiliza CRF++.

7.4.3. WEKA

WEKA[8] (*Waikato Environment for Knowledge Analysis*) es un *toolkit* utilizado para la minería de datos, que permite analizar diversos algoritmos y técnicas utilizadas para el análisis de datos. En nuestro caso resulta útil para el aprendizaje automático sobre un corpus. Para poder utilizar WEKA es necesario que la información a analizar tenga un formato que la herramienta pueda entender, este formato es llamado ARFF y se detalla más adelante.

WEKA es una distribución de software libre desarrollada en Java y está formada de diversas técnicas de pre procesado, clasificación, agrupamiento, asociación y visualización contenidas en paquetes *open source*. Estos paquetes pueden integrarse con un proyecto para analizar datos o es posible utilizar una interfaz gráfica de usuario para acceder a las técnicas mencionadas anteriormente. En particular en nuestro proyecto utilizamos dicha interfaz aplicando el algoritmo J48 a nuestro corpus, el cual es una implementación del ya mencionado algoritmo C4.5. Una particularidad de Weka es que todas las implementaciones de árboles de decisión no permiten tener atributos de tipo *String*.

ARFF

Un archivo ARFF es un archivo en ASCII que describe una lista de instancias que comparten un conjunto de atributos.

Estos archivos se dividen en dos secciones:

1. La información de cabecera
2. Los datos

La cabecera contiene el nombre de la relación (@*relation* <nombre-de-la-relación>), una lista de atributos y sus tipos (de la forma @*attribute* <nombre-del-atributo> tipo).

Los tipos soportados en Weka son:

- *numeric*: expresa números reales
- *integer*: expresa números enteros
- *date*: expresa fechas, se debe indicar el formato.
- *string*: expresa cadenas de texto
- enumerado: se definen proveyendo su especificación como una lista de posibles valores entre llaves, separados por coma (ejemplo: @*attribute* tiempo {soleado,nublado,lluvioso})

En la sección de datos se declaran los datos que componen la relación separando entre comas los atributos y con saltos de líneas las relaciones.

[8] http://www.cs.waikato.ac.nz/ml/weka/

La Figura 19 muestra un ejemplo de un archivo ARFF que representa los atributos id, *token*, lema, CPOSTAG, POSTAG, dependencia, es_evento y clase real de una oración del corpus.

```
@relation factividad

@attribute id INTEGER
@attribute palabra STRING
@attribute lema STRING
@attribute cpostag {a,r,d,n,v,p,c,i,s,f,z,w}
@attribute postag STRING
@attribute dependencia INTEGER
@attribute es_evento {0,1}
@attribute class {O,Brealizado, Bno_realizado, Bindef}

@data
1,A,a,s,SPS00,0,0,O
2,ver,ver,v,VMN0000,1,0,O
3,si,si,c,CS,0,0,O
4,conseguimos,conseguir,v,VMIS1P0,3,1,Bindef
5,avanzar,avanzar,v,VMN0000,4,1,Bindef
6,en,en,s,SPS00,5,0,O
7,este,este,d,DD0MS0,8,0,O
8,tema,tema,n,NCMS000,6,0,O
9,sin,sin,s,SPS00,5,0,O
10,disputas,disputa,n,NCFP000,9,0,O
11,ni,ni,c,CC,10,0,O
12,demagogias,demagogia,n,NCFP000,11,0,O
13,.,.,f,Fp,12,0,O
```

Figura 19 - Ejemplo de un archivo ARFF

7.5. Implementación

La implementación tanto de la línea base como del sistema en sí se realizó utilizando el lenguaje Python 3 y las herramientas ya mencionadas en este capítulo.

Se trabajó en Ubuntu 12.10 con *scripts* que son ejecutados desde la consola.

7.5.1. Línea base

Para la implementación de la línea base primero determinamos el conjunto de palabras que van a indicar negación (Tabla 20). Estas, como ya se mencionó, son los índices de polaridad obtenidos del corpus (*no*, *nada*) y las palabras *nunca* y *sin*.

no
nunca
nada
sin

Tabla 20 - Lista de palabras que indican negación

La implementación de la línea base consiste de tres *scripts*, esta estructura se muestra en el diagrama de la Figura 20:

1. *mainBase.py*: este lee los archivos del corpus (corpus en texto plano) y llama al *script baseline.py* y luego al *compareBase.py*. Por último el *compareBase.py* devuelve la matriz de confusión la cual se imprime en pantalla.

2. *baseline.py*: este *script* obtiene de Freeling el etiquetado POS de cada oración y para cada evento, en nuestro caso restringiéndonos a los verbos (CPOSTAG igual a v), le asigna un valor de factividad. Este valor de factividad consiste en:

    i) Si el verbo está en futuro (si la tercera columna del POSTAG es F) etiqueta al evento como **indef**.

    ii) Si existe una palabra dentro de la oración antes del evento que pertenece a la lista de palabras negadas se etiqueta al evento como **no_realizado**. Para esto se debe llevar una lista de palabras ya visitadas.

    iii) En otro caso se etiqueta al evento como **realizado**.

    Este *script* arma una matriz de 3 dimensiones, donde la primera dimensión es la oración, la segunda es el evento y la tercera la factividad etiquetada.

3. *compareBase.py*: Dado el arreglo de eventos devuelto por *baseline.py* clasificados y el archivo original etiquetado por una de las anotadoras (corpus en formato XML) este *script* compara ambas anotaciones y arma la matriz de confusión, la cual es devuelta al *mainBase.py* para imprimir.

Cabe destacar que se debieron realizar algunos procesamientos aquí dado que el archivo original está en XML y dado que consideramos en la línea base sólo los eventos que son verbos y donde un evento consiste de un solo *token* (originalmente podían haber muchos *tokens* en un evento).

Esta implementación de la línea base obtuvo una exactitud de 68,5%.

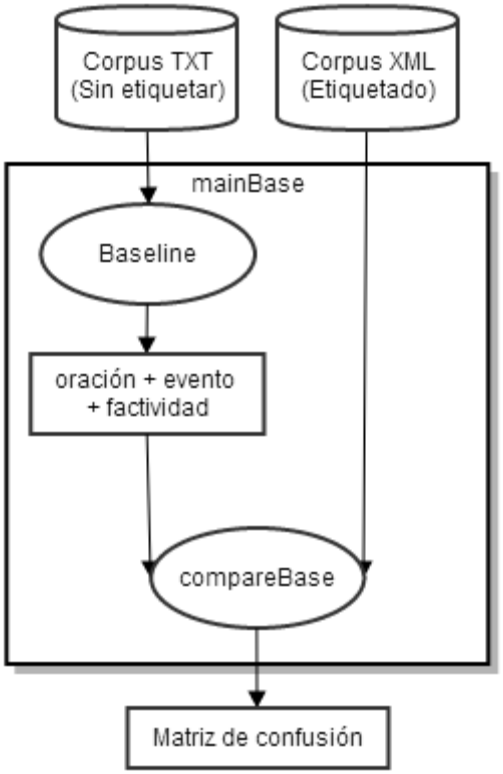

Figura 20 – Diagrama de la línea base

7.5.2. Sistema

Lo primero realizado fue la recolección de palabras que indican negación, modalidad e implicaciones (se encuentran en el anexo en la sección B). La implementación del sistema consiste de un pre procesamiento de los archivos del corpus para transformarlos en formato CONLL luego se ejecuta el *toolkit* correspondiente (CRF++ o Yamcha) y finalmente se determina el desempeño del sistema. A excepción de los *toolkits* y las herramientas ya mencionadas, todo el resto son *scripts* implementados en Python:

1) *mainScript.py*: este *script* recibe como entrada la versión de prueba en la que nos encontramos y el corpus a procesar (entrenamiento, desarrollo o testeo). Para cada archivo del corpus (del corpus en texto plano sin etiquetar) primero llama al *mainDependency.py* quien devuelve el conjunto de eventos del archivo (según nuestra definición) y también transforma el archivo en formato CONLL básico devuelto por Freeling (sin relación de dependencias). Luego se ejecuta Maltparser con este archivo CONLL base para determinar las dependencias. A continuación se llama al *script* mainCompare.py quien determina el valor de factividad de los eventos determinados en el corpus acorde a lo etiquetado por las anotadoras. Finalmente se llama nuevamente al *script mainDependency.py* quien arma los archivos CONLL completos con la etiqueta real al final de cada *token* para ser usado por CRF++ y Yamcha. La Figura 21 muestra el diagrama para este *script*.

2) *mainDependency.py*: este *script* consta de dos métodos importantes:

   1. *fase1*: utiliza el etiquetador POS de Freeling para determinar los atributos 1 a 5 y escribir un archivo en formato CONLL (sin dependencias) de estos valores. A su vez determina cuales son los verbos (si el CPOSTAG es v) que serán luego pasados al comparador para que determine el valor de factividad de este evento en los archivos etiquetados.

   2. *fase3*: dado el conjunto de eventos con sus valores de factividad anotados por las anotadoras y el archivo generado por Maltparser, arma el archivo final a ejecutar en CRF++ y Yamcha con el formato CONLL y donde la última columna de cada *token* es la etiqueta real del evento o O si no se trata de un evento.
      Este método difiere dependiendo de la versión en la que nos encontramos del sistema, donde en particular en algunas versiones debemos armar un árbol de dependencias para saber si el evento tiene algún hijo o antecesor perteneciente a alguna de las listas. Otro caso particular es la regla que busca la polaridad basándose en la implicación de su padre o abuelo y su polaridad.

3) *mainCompare.py*: Dado el archivo original en formato XML (ya etiquetado por las anotadoras) y el conjunto de eventos por oración, determina la etiqueta asignada por la anotadora y devuelve una matriz de 3 dimensiones donde la primera dimensión es la oración, la segunda es el evento y la tercera la factividad.

4) *measureScript.py*: *script* que dada la salida de los *toolkits* (CRF++ o Yamcha) compara las dos últimas etiquetas de cada *token* (la etiqueta real y la predicha) y arma la matriz de confusión. A su vez, también discriminamos los resultados según los distintos modos y formas verbales.

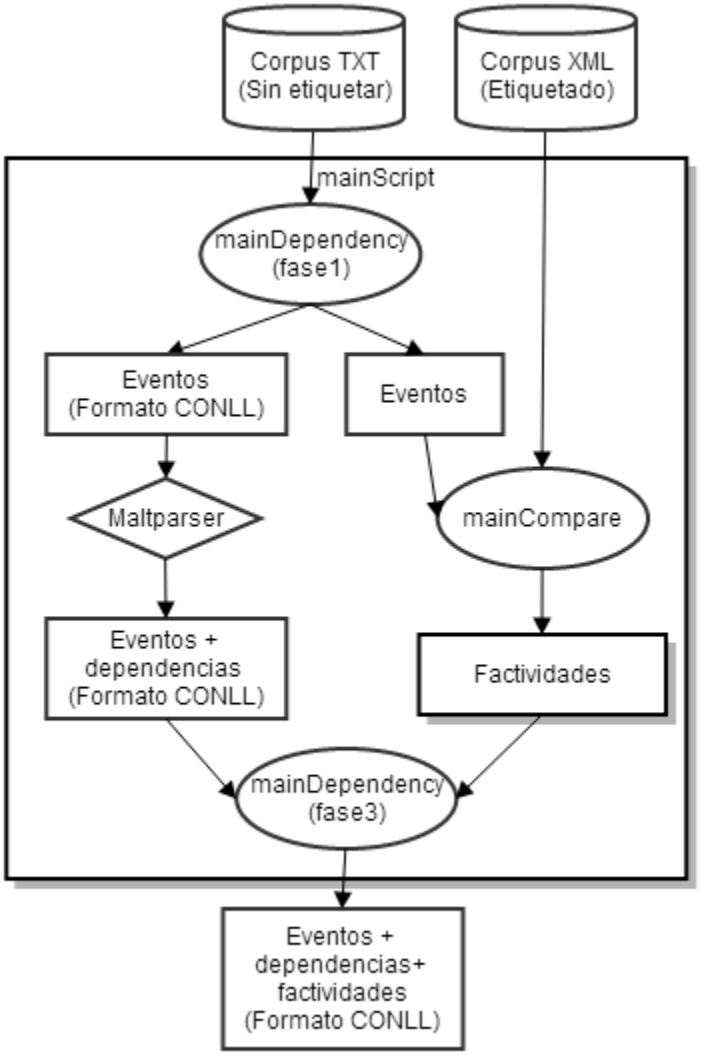

Figura 21 – Diagrama de la construcción de los archivos en formato CONLL

Para la implementación usando *Decision Tree* (en particular Weka) se debió pasar los archivos en formato CONLL a formato ARFF. También surgió el problema de que los árboles de decisión en Weka no aceptan *Strings* como atributos, tipo que se usaba tanto para las palabras, los lemas, los POSTAGs y las relaciones de dependencia. Por esto se debió llevar una lista de todas los *tokens*, todos los lemas, todos los POSTAGs y todas las relaciones de dependencia en el corpus para poder ser usados como enumerados donde otra restricción es que los valores de los enumerados no pueden estar repetidos. Para lograr construir archivos ARFF a partir de archivos en formato CONLL se crearon dos *scripts*:

- *toArff.py*: este *script* tiene varias funciones:
    - *stringsToNominals*: que recorre todos los archivos del corpus en formato CONLL y obtiene una lista de todas los *tokens*, otra de los lemas, otra de los POSTAGs y otra de las relaciones de dependencias (todas ellas sin elementos repetidos)

- *attributes*: que dada la versión en la que nos encontramos arma la lista de atributos para poner en el cabezal del archivo ARFF. En particular debe recorrer las listas armadas en *stringsToNominals* para ponerlas como un enumerado.

- *data*: recorre un archivo pasado como parámetro (en formato CONLL) y reemplaza los tabuladores por comas. Para esta función se asume que se le pasa el archivo con la misma cantidad de atributos que los formados en el cabezal.

- *transformToArff.py*: esta función recibe en la línea de comando la versión que se está corriendo y llama al script *toArff.py* para armar los archivos de entrenamiento y testeo en formato ARFF para la versión determinada.

La Figura 22 muestra el proceso luego de ejecutado el *mainScript.py* para pasar por los distintos *toolkits*.

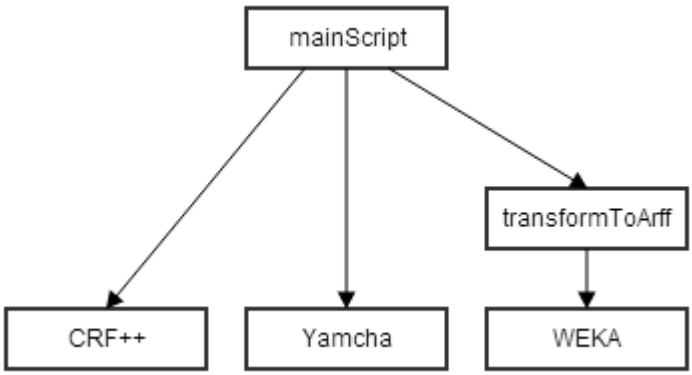

Figura 22 – Diagrama de los *toolkits*

Capítulo 8

Evaluación de la herramienta

En este capítulo se muestran las métricas utilizadas para medir el desempeño del sistema. Se dividió el corpus entre entrenamiento, desarrollo y testeo. Luego se definió un procedimiento iterativo con el corpus de entrenamiento y desarrollo para determinar los atributos finales del sistema.

Se presentan en este capítulo el procedimiento iterativo y los resultados obtenidos en cada versión (dependiente de los atributos) y en cada *toolkit* (CRF++ y Yamcha). También se presentan los resultados de una versión de Weka donde se observa que su rendimiento no es aceptable y por esto se decidió no continuar con las pruebas. Por último se muestra el resultado final del corpus de testeo, las curvas de aprendizaje para el corpus de desarrollo y de testeo y algunos ejemplos de errores de los clasificadores.

8.1. Métricas de desempeño

8.1.1. Exactitud de Factividad

Dado que nuestro interés se focaliza en clasificar la factividad de los eventos y no los eventos en sí se decidió tener una medida que muestre la proporción de los eventos con factividad clasificados correctamente. A su vez al considerar solo los verbos y tener un atributo que especifica que estamos dentro de nuestra definición de evento, la exactitud general supera siempre el 97%. Por esto decidimos no considerar los *tokens* que no son eventos o son clasificados como no eventos al calcular la exactitud de factividad.

La siguiente tabla muestra una matriz de confusión como las utilizadas en este proyecto, donde $Tfactividad_i$ indica los eventos clasificados con $factividad_i$ que realmente pertenecen a la $factividad_i$ y los $Ffactividad_i$,$factividad_j$, son los eventos clasificados incorrectamente con $factividad_i$ y que realmente eran de tipo $factividad_j$. Donde se observa que además de los valores de factividad ya mencionados tenemos el valor no_evento que indica que el *token* no es un evento según nuestras restricciones.

Real\Predicha	**realizado**	**no_realizado**	**Indef**	**no_evento**
realizado	Trealizado	Fno_realizado, realizado	Findef, realizado	Fno_evento, realizado
no_realizado	Frealizado, no_realizado	Tno_realizado	Findef, no_realizado	Fno_evento, no_realizado
indef	Frealizado, indef	Fno_realizado, indef	Tindef	Fno_evento, indef
no_evento	Frealizado, no_evento	Fno_realizado, no_evento	Findef, no_evento	Tno_evento

Tabla 21 - Ejemplo de matriz de confusión para la factividad

Definimos la exactitud de factividad como:

$$\frac{Trealizado + Tno\_realizado + Tindef}{\sum_{i=realizado}^{indef} \sum_{j=realizado}^{indef} F_{i,j} + Trealizado + Tno\_realizado + Tindef}$$

Es decir, se calcula la exactitud de una matriz de confusión restringida solo a los valores de factividad, sin tener en cuenta los valores asociados a los *tokens* que tanto eran o fueron clasificados como no_evento.

8.2. Entrenamiento y Evaluación

Para aplicar aprendizaje supervisado optamos por dividir el corpus en tres subconjuntos disjuntos.
El primer corpus llamado **corpus de entrenamiento**, este es un 70% del corpus total y es utilizado con el fin de inferir conocimiento útil para el sistema para luego poder clasificar una instancia desconocida.

Luego tenemos un segundo corpus llamado **corpus de desarrollo**, este es un 15% del corpus total y es usado con diversos fines, principalmente para medir qué tan bien "aprende" nuestro sistema. Además se utiliza con el fin de sacar conclusiones sobre el aprendizaje y analizar la clasificación que éste realiza con el fin de ajustar los parámetros de los métodos de aprendizaje y los atributos utilizados para mejorar el desempeño del sistema.

Por último tenemos un tercer corpus, llamado **corpus de testeo**, este es un 15% del corpus total y se utiliza con el fin de medir el desempeño del sistema, en esta instancia ya no se ajustan parámetros y atributos, es un corpus que el sistema no ha procesado antes.

En nuestro caso, dado que tenemos un total de 573 oraciones, separamos el corpus de entrenamiento, desarrollo y testeo en 401, 86 y 86 oraciones respectivamente. Dentro de cada corpus la cantidad de eventos con cada factividad se muestra en la Tabla 22.

Factividad\Corpus	**Entrenamiento**	**Desarrollo**	**Testeo**	**Total**
Realizado	952	240	200	1392
No_realizado	74	31	16	121
Indef	436	78	53	567
Total	1462	349	269	2080

Tabla 22 - Factividades en cada corpus (entrenamiento, desarrollo y testeo)

8.3. Weka

Se realizaron varias pruebas utilizando la herramienta Weka sin generar resultados mayores al 54,8% de exactitud, la Tabla 23 muestra la precisión, recuperación y Medida F del mejor resultado. Se nota que la exactitud es muy inferior a la línea base (es 13,7% inferior).

Se observa que nunca se clasifica a un evento como no_realizado. Esto se visualiza en el árbol de decisión formado por Weka, el mostrado en la Figura 23, que no contiene ninguna hoja para el caso de los eventos no realizados.

Cabe mencionar que esta prueba no se realizó con la totalidad de los atributos, pero si con algunos básicos, pero dado que los resultados no eran aceptables se decidió no continuar con las pruebas.

	Precisión	**Recuperación**	**Medida F**
Realizado	59,8%	100%	74,9%
No_realizado	0%	0%	0%
Indef	100%	11,8%	21,2%

Tabla 23 - Precisión, Recuperación y Medida F de Weka

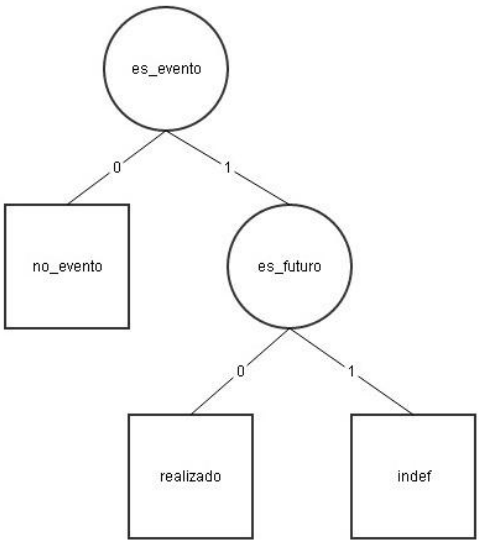

Figura 23 - Árbol de decisión formado por Weka

Luego de realizadas estas pruebas concluimos, como ya sospechábamos, que los árboles de decisión no son adecuados para este tipo de problemas.

8.4. Experimentos

En una primera instancia se decidió separar los atributos en grupos, donde los atributos 1 a 13 son los básicos, y a los que se les van agregando atributos de a dos, salvo por el caso de los atributos 24 a 27 que consideramos que pertenecen a un mismo grupo y el 28 que por ser el último que queda se agrega solo.

Exp	1-13	14-15	16-17	18-19	20-21	22-23	24-27	28	CRF++	Yamcha
Exp 1	X								83,5%	83,3%
Exp 2	X	X							84,1%	82,7%
Exp 3	X	X	X						83,1%	82,7%
Exp 4	X	X	X	X					83,4%	83%
Exp 5	X	X	X	X	X				83,2%	83,6%
Exp 6	X	X	X	X	X	X			83,8%	82,8%
Exp 7	X	X	X	X	X	X	X		83,2%	83,1%
Exp 8	X	X	X	X	X	X	X	X	83,2%	83,4%

Tabla 24 - Tabla de experimentos y atributos con sus respectivos valores de exactitud de factividad.

La Tabla 24 muestra los atributos usados en cada experimento y el resultado obtenido de dicho experimento. Se observa que en todo momento los resultados de CRF++ y Yamcha son cercanos, la máxima diferencia se da en el experimento 2 donde CRF++ supera por 1,4% a Yamcha. Yamcha solo supera a CRF++ en dos ocasiones: en el experimento 5 (que lo supera por 0,4%) y en el experimento 8 (que lo supera por 0,2%).

Por último se observa que los valores de CRF++ oscilan entre 83,10% y el 84,10%, mientras que Yamcha lo hace entre 82,7% y 83,6%, siempre superando la exactitud de la línea base.

8.4.1. Implicaciones

Luego de observar que los atributos que consideraban las implicaciones (experimentos 7 y 8) no parecen ayudar a mejorar el desempeño del sistema, decidimos verificar cuántos verbos pertenecientes a cada lista de implicaciones existen en cada corpus, se muestra en la Tabla 25 la cantidad de implicaciones de cada tipo que se encuentra en el corpus de entrenamiento y desarrollo. Se observa que el total de las implicaciones no coincide con la suma de cada tipo de implicación, dado que una misma palabra puede pertenecer a más de una lista de implicaciones, esto hace claramente más complicado e ineficiente nuestro algoritmo, dado que solo considerará el primero que aparezca en el algoritmo. La cantidad pequeña de palabras de cada implicación justifica la poca importancia de estos atributos en el sistema.

Corpus\Implicación	IP (+)(+)	IP (+)(-)	IP (-)(+)	IP (-)(-)	Total
Entrenamiento	87	52	0	79	164
Desarrollo	14	19	0	21	36

Tabla 25 - Cantidad de implicaciones en el corpus de entrenamiento y desarrollo

Se observa que tanto en el corpus de entrenamiento como en el de desarrollo no existen implicaciones de tipo (-)(+).

8.5. Versión Final

8.5.1. Selección de los atributos y ventanas

Luego de realizadas los experimentos anteriores decidimos intentar mejorar el aprendizaje, para esto realizamos diferentes pruebas sobre el corpus de desarrollo utilizando los *toolkits* Yamcha y CRF++. Esto lo hicimos quitando atributos y/o variando el tamaño de las ventanas de los atributos. Estas pruebas fueron realizadas en paralelo con ambos *toolkits*, con el fin de obtener un buen resultado para los dos casos y poder compararlos entre sí. Al tomar decisiones sobre los parámetros que afectan los atributos siempre nos quedamos con un resultado que no empeore ninguno de los *toolkits*. Es decir, nos quedamos con una prueba que o bien mejora para ambos *toolkits*, o bien mejora para uno y el otro permanece igual.

A continuación se detalla el proceso utilizado y conclusiones obtenidas en el mismo.

Para comenzar tomamos los 28 atributos con los valores por defecto para los *toolkits* ambos con una ventana de [-2,2] para cada uno de los atributos.

La primera prueba que realizamos fue ir quitando uno a uno los atributos utilizando una regla básica de descarte, si el quitar un atributo mejora el desempeño del sistema lo sacamos, en caso contrario queda en el sistema.

Mediante este proceso obtuvimos como resultado los atributos que convenía sacar para nuestro corpus de desarrollo, dado que empeoraban o no aportaban a la clasificación, estos son:

- **1. id:** Este atributo tiene sentido que no aporte dado que el lugar dentro de la oración por sí mismo no parece aportar nada. A su vez este atributo junto con el id del padre es utilizado al contemplar el árbol de dependencias en busca de palabras que indiquen negación o posibilidad.

- **4. CPOSTAG:** El CPOSTAG está incluido dentro del POSTAG (es el primer carácter del POSTAG) por lo que tiene sentido que este no sea necesario. A su vez, restringimos los eventos a los verbos (CPOSTAG igual a v), y ya existe una columna que indica si el *token* es evento en nuestra restricción por lo que tener un CPOSTAG en v no debería aportar y podría incluso confundir al sistema ya que intentaría etiquetar todos los verbos como eventos.

- **6. Id del padre:** Al igual que el atributo 1 (Id del *token*) este atributo en sí no aporta nada, y ya es utilizado en los atributos que contemplan el árbol de dependencias.

- **11. Está en presente:** Al observar las matrices de confusión antes y después de sacar este atributo nos damos cuenta de que dos eventos que estaban siendo clasificados como realizados pero eran no realizados pasan a clasificarse correctamente. Esto se puede deber a que, como muestra la Tabla 26, la mayoría de los eventos en presente son realizados, por lo que el sistema podría tender a etiquetar todos los eventos en presente como realizados, cuando este no es el caso.

- **15. Es indicativo:** Nuevamente observando las matrices de confusión antes y después de sacar este atributo nos damos cuenta que dos eventos que estaban siendo clasificados como realizados pero eran no realizados se pasan a clasificar correctamente. Esto nuevamente puede deberse a que, como se observa en la Tabla 27, la cantidad de eventos en modo indicativo que son realizados es muy superior a los no realizados, lo cual llevaría al sistema a clasificar incorrectamente a los eventos no realizados como realizados.

- **24. Pertenece a la lista de implicaciones + +:** Al observar las matrices de confusión antes y después de sacar este atributo se nota que un evento que se estaba clasificando incorrectamente como realizado pero era indefinido, pasa a clasificarse correctamente. Esto podría deberse a que, dado que la proporción de eventos pertenecientes a la lista de implicaciones + + de factividad realizado es muy superior a las otras factividades (Tabla 28), el sistema puede tender a clasificar dichos eventos como realizados. Además se observa de la Tabla 25 que el corpus de desarrollo solo contiene 14 eventos con implicación + +.

- **26. Pertenece a la lista de implicaciones - +:** Como se observó en la Tabla 25 no existen implicaciones de tipo - + ni en el corpus de entrenamiento ni en el de desarrollo, por lo cual es razonable que este atributo no ayude a mejorar el sistema.

Corpus\Factividad	Realizado	No Realizado	Indef
Desarrollo	116	23	26
Entrenamiento	403	37	88

Tabla 26 - Factividades por corpus (entrenamiento y desarrollo) para los eventos en tiempo presente

Corpus\Factividad	Realizado	No Realizado	Indef
Desarrollo	186	23	27
Entrenamiento	672	47	166

Tabla 27 - Factividades por corpus (entrenamiento y desarrollo) para los eventos en modo indicativo

Corpus\Factividad	Realizado	No Realizado	Indef
Desarrollo	10	1	3
Entrenamiento	56	2	27

Tabla 28 - Factividades por corpus (entrenamiento y desarrollo) para los eventos con implicación ++

En general se observa que todos estos atributos mencionados tiene sentido que se remuevan porque su información es irrelevante, están contemplados en otro atributo o confunden al sistema.

Se muestra en las Tablas 29 y 30 las matrices de confusión para CRF++ y Yamcha para esta versión. La Tabla 31 muestra la exactitud de factividad para CRF++ y Yamcha.

Clase real\clase predicha	Realizado	No_realizado	Indef
Realizado	223	1	14
No_realizado	7	20	3
Indef	21	1	54

Tabla 29 - Matriz de confusión para la versión final de atributos para CRF++

Clase real\clase predicha	Realizado	No_realizado	Indef
Realizado	222	2	12
No_realizado	8	21	2
Indef	20	1	55

Tabla 30 - Matriz de confusión para la versión final de atributos para Yamcha

Toolkit\Resultado	Exactitud de factividad
CRF++	86,3%
Yamcha	86,9%

Tabla 31 - Exactitud de Factividad para CRF++ y Yamcha para la versión final de atributos

Se observa que ambos *toolkits* mejoran con respecto a las versiones 1 a 8, y que Yamcha clasifica 0,6% mejor que CRF++.

Luego de encontrar la mejor combinación de atributos, decidimos para estos atributos investigar cuál era la mejor ventana, para ello se realizó una segunda etapa de pruebas. Partiendo de los resultados anteriores se probó con tres ventanas distintas para cada atributo además de la ya probada [-2,2] (valor por defecto):

1- Ventana [0]
2- Ventana [-1,1]
3- Ventana [-3,3]

Nuevamente se fue probando uno a uno los atributos con las ventanas mencionadas utilizando la siguiente regla de descarte, si alguna de las ventanas mejora el desempeño del sistema nos quedamos con la ventana de mayor mejora, en caso contrario dejamos el valor por defecto [-2,2].

Mediante este proceso se obtuvo como resultado que únicamente en el atributo 19 “tiene un hijo en la lista de posibilidades” es conveniente modificar la ventana al tamaño [-1,1].

En esta etapa se observó que es muy difícil encontrar una ventana que mejore para ambos *toolkits*, por lo general si uno mejoraba el otro empeoraba, con lo cual nos veíamos obligados a quedarnos con la ventana por defecto [-2,2].

Cabe destacar que es posible realizar un número muy grande de pruebas con las combinaciones de ventanas sobre los atributos, nosotros realizamos las que consideramos más relevantes y que podrían mejorar nuestro sistema.

La matriz de confusión para CRF++ para este caso se muestra en la Tabla 32, donde la exactitud de factividad es de 86,6%, Yamcha por su lado se mantiene igual (Tabla 30).

Clase real\clase predicha	Realizado	No_realizado	Indef
Realizado	224	1	13
No_realizado	7	20	3
Indef	21	1	54

Tabla 32 - Matriz de confusión para la versión final de CRF++

Se observa que acortando la ventana un evento que era realizado pero estaba siendo clasificado como indefinido, pasa a estar bien clasificado. A pesar de aumentar la exactitud de CRF++ esta sigue siendo inferior a la de Yamcha, aunque solo por 0,3%.

Finalmente los atributos finales, junto con sus ventanas son los siguientes:

2. Token [-2,2]
3. Lema [-2,2]
5. POSTAG [-2,2]
7. Relación con el padre [-2,2]
8. Es evento [-2,2]
9. Está en futuro [-2,2]
10. Está en pasado [-2,2]
12. Modo [-2,2]
13. Tiempo [-2,2]
14. Es condicional [-2,2]
16. Pertenece a la lista de negaciones [-2,2]
17. Pertenece a la lista de posibilidades [-2,2]
18. Tiene un hijo en la lista de negaciones [-2,2]
19. Tiene un hijo en la lista de posibilidades [-1,1]
20. Tiene padre o abuelo que pertenece a la lista de negaciones [-2,2]
21. Tiene padre o abuelo que pertenece a la lista de posibilidades [-2,2]
22. Tiene hijo de antecesor de menor id que pertenece a la lista de negaciones [-2,2]
23. Tiene hijo de antecesor de menor id que pertenece a la lista de posibilidades [-2,2]
25. Pertenece a la lista de implicaciones + - [-2,2]
27. Pertenece a la lista de implicaciones - - [-2,2]
28. Implicación [-2,2]

Si comparamos el primer experimento realizado (atributos básicos) con esta última versión (Tabla 33), se puede observar que si bien se tienen 8 atributos más, cada uno de estos más complejos que cualquiera de los atributos básicos, se consigue una mejora considerable de un 3,1% en CRF++ y 3,6% en Yamcha.

Toolkit\Experimento	Experimento 1	Experimento Final
CRF++	83,5%	86,6%
Yamcha	83,3%	86,9%

Tabla 33 - Comparación entre experimento inicial y final en el corpus de desarrollo.

8.5.2. Curva de Aprendizaje

La curva de aprendizaje es un gráfico que muestra el éxito obtenido durante el aprendizaje a medida que se agranda el corpus. Es interesante utilizar esta métrica, ya que dependiendo de cuan empinada sea la curva podemos saber si en la tarea de aprendizaje con un corpus chico se aprende mucho (curva empinada) o, de lo contrario, con un corpus chico se aprende poco (curva poco empinada). Además podemos saber en qué momento el agregar más datos ya no genera una mejora en el aprendizaje (llanura).

En nuestro caso utilizamos este gráfico para ver cómo se comporta el sistema dependiendo de la cantidad de oraciones en el corpus de entrenamiento. Para ello graficamos la exactitud de la factividad en función del porcentaje de corpus de entrenamiento, partiendo con un 20% de dicho corpus y aumentando de a 20% hasta llegar al 100%.

Para esto se dividió el corpus de entrenamiento en cinco partes, cada una de 80 oraciones (20%). Luego se fue agregando de a uno estos subcorpus para ir viendo que tanto mejoraba la clasificación al aumentar la cantidad de corpus. Este mecanismo se realizó tanto para el corpus de desarrollo, como para el de testeo.

8.5.2.1. Corpus de desarrollo

La Tabla 34 muestra las exactitudes de factividad para CRF++ y Yamcha para cada subconjunto de corpus de desarrollo. Los Gráficos 1 y 2 muestran las curvas de aprendizaje para CRF++ y Yamcha respectivamente.

Toolkit\Corpus	**20%**	**40%**	**60%**	**80%**	**100%**
CRF++	80,8%	82,9%	85,5%	85,4%	86,6%
Yamcha	79,9%	75,3%	85,75%	84%	86,9%

Tabla 34 - Tabla de Curva de Aprendizaje para el corpus de desarrollo

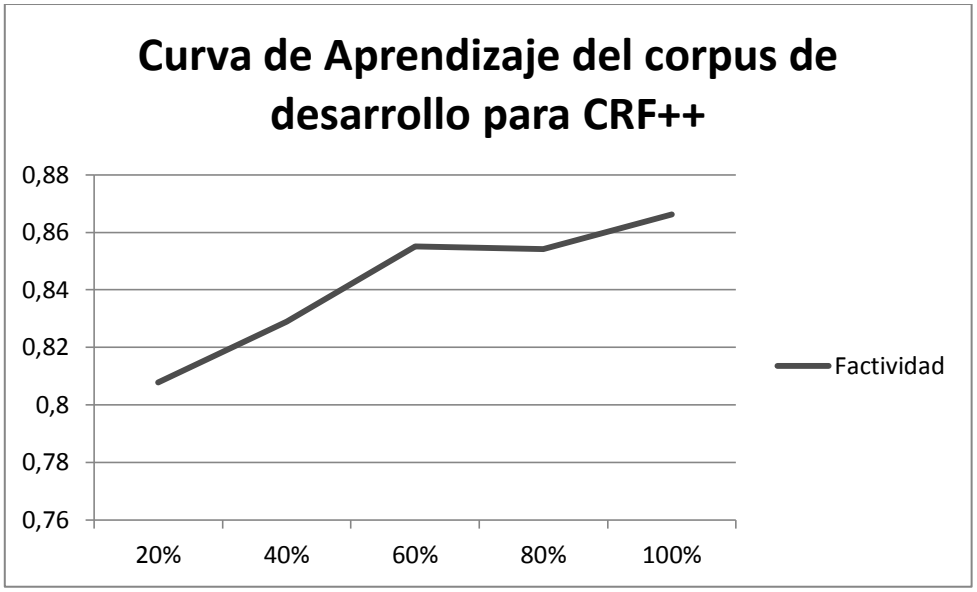

Gráfico 1 - Curva de aprendizaje del corpus de desarrollo para CRF++

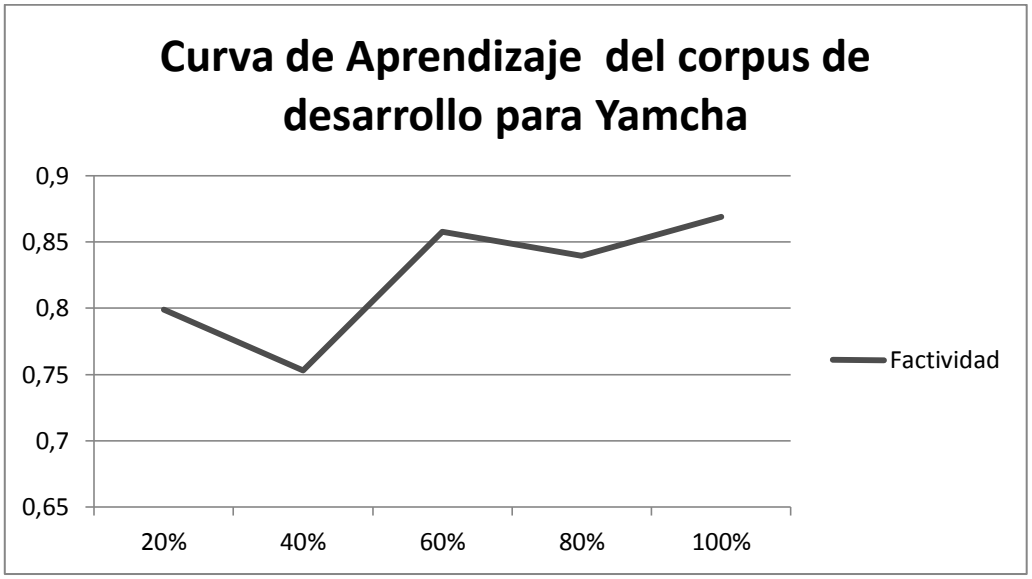

Gráfico 2 - Curva de aprendizaje del corpus de desarrollo para Yamcha

Ambas curvas de aprendizaje parecen indicar que no se ha alcanzado la llanura, por lo que aumentar el corpus de entrenamiento haría mejorar el resultado del sistema.

8.5.2.2. Corpus de testeo

La Tabla 35 muestra la exactitud de factividad para CRF++ y Yamcha para cada subconjunto del corpus de testeo. Los Gráficos 3 y 4 muestran las curvas de aprendizaje para CRF++ y Yamcha respectivamente.

Toolkit\corpus	20%	40%	60%	80%	100%
CRF++	82,4%	85,9%	88,5%	87,4%	85,1%
Yamcha	83,5%	86,5%	88,5%	89,7%	87,4%

Tabla 35 - Tabla de Curva de Aprendizaje para el corpus de testeo

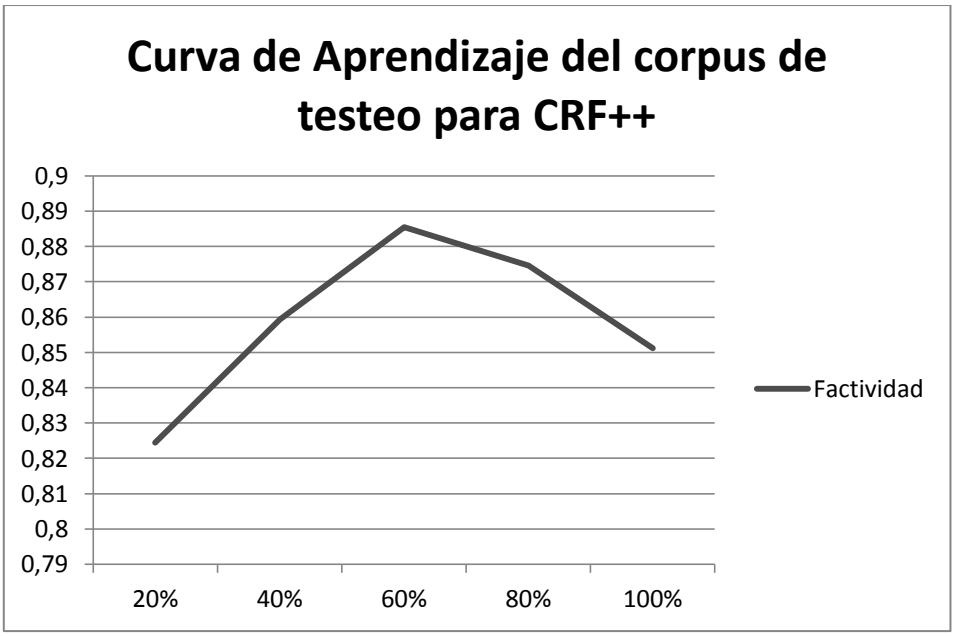

Gráfico 3- Curva de aprendizaje del corpus de testeo para CRF++

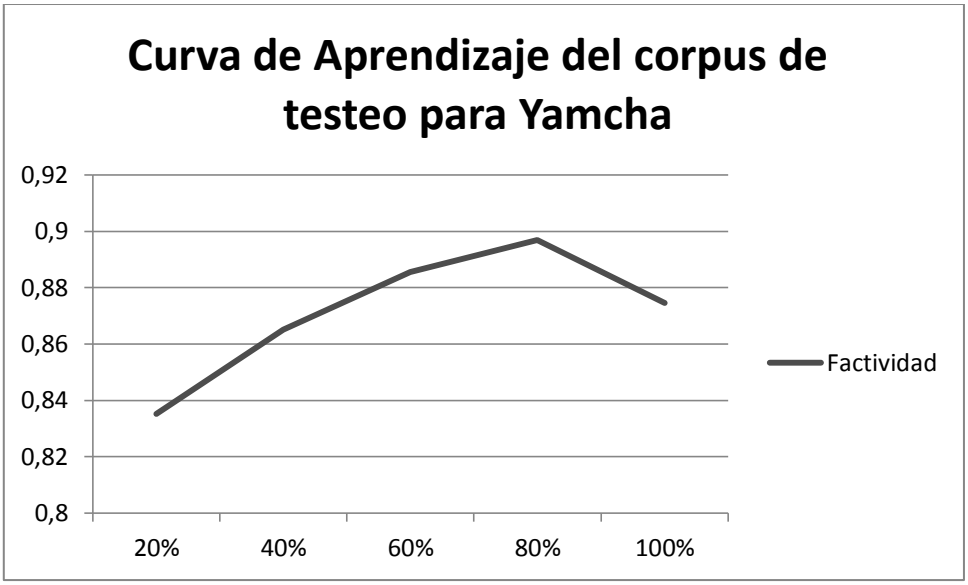

Gráfico 4 - Curva de aprendizaje del corpus de testeo para Yamcha

A pesar de que las curvas de aprendizaje del corpus de desarrollo parecían indicar que aumentar el corpus de entrenamiento mejoraría el rendimiento del sistema, la curva de aprendizaje del corpus de testeo no parece indicar lo mismo. Esto podría deberse a un sobreajuste producido porque los atributos determinados se ajustaban al corpus de desarrollo, pero no así al corpus de testeo. Cabe destacar que el corpus de testeo está conformado en su totalidad por los textos del corpus extendido, mientras que el de entrenamiento y desarrollo contienen una mezcla de ambos corpus. Esto podría implicar que el sistema se ajusta mejor al corpus inicial, y no así al corpus extendido. A su vez, el corpus de testeo, al corresponder al corpus extendido, está anotado por las tutoras y el corpus de entrenamiento casi en su totalidad está anotado por las estudiantes de lingüística. Por último se observa que, en la curva de aprendizaje del corpus de testeo, la diferencia entre el resultado máximo alcanzado y el final difiere solo en 2,3% en ambos *toolkits* lo cual no parece significativo en un resultado mayor al 85%.

8.5.3. Corpus de Testeo

Por último mostramos las matrices de confusión para CRF++ y Yamcha para el corpus de testeo en las Tablas 36 y 37. Las Tablas 38 y 39 muestran la precisión, recuperación y Medida F para cada factividad. La Tabla 40 muestra la exactitud de factividad para cada *toolkit*.

Clase real\clase predicha	**Realizado**	**No_realizado**	**Indef**
Realizado	178	0	15
No_realizado	1	13	2
Indef	19	2	32

Tabla 36 - Matriz de confusión del corpus de testeo para CRF++

Clase real\clase predicha	Realizado	No_realizado	Indef
Realizado	183	0	11
No_realizado	2	12	2
Indef	16	2	35

Tabla 37 - Matriz de confusión del corpus de testeo para Yamcha

	Precisión	Recuperación	Medida F
Realizado	89,9%	92,2%	91%
No_realizado	86,7%	81,25%	83,9%
indef	65,3%	60,4%	62,7%

Tabla 38 - Precisión, recuperación y Medida F para el corpus de testeo de CRF++

	Precisión	Recuperación	Medida F
Realizado	91%	94,3%	92,6%
No_realizado	85,7%	75%	80%
indef	72,9%	66%	69,3%

Tabla 39 - Precisión, recuperación y Medida F para el corpus de testeo de Yamcha

Toolkit	Exactitud Factividad
CRF++	85,1%
Yamcha	87,4%

Tabla 40 – Exactitud de factividad para el corpus de testeo para CRF++ y Yamcha

Como conclusión de estos resultados podemos decir que se obtuvieron resultados satisfactorios comparados con la línea base (68,5%), aunque no se alcanzó la línea tope (90,4%). También se observa que Yamcha clasifica mejor en un 2,3%. En particular, discriminando según las factividades se observa que Yamcha clasifica mejor los eventos realizados e indefinidos, mientras que CRF++ clasifica mejor los eventos no realizados. Se observa que a pesar de que los eventos no realizados eran los de menor cantidad de datos en el corpus, su clasificación fue mejor que los indefinidos, incluso a pesar de que la cantidad de eventos indefinidos supera por más de cuatro veces la cantidad de eventos no realizados. Claramente se debe mejorar en la clasificación de eventos indefinidos, ya que su clasificación es superada en más de un 10% por los eventos no realizados y en más de un 20% por los eventos realizados (en ambos clasificadores).

También es importante notar que, dado que el objetivo del trabajo era comparar los dos modelos CRF y SVM, hubo atributos que no se sacaron porque, a pesar de mejorar el rendimiento del sistema con un modelo, empeoraba el del otro, lo mismo sucedía con las ventanas. Pero, en caso de querer focalizarnos en un modelo en particular, el resultado podría haber sido mejor.

8.5.3.1. Resultados discriminados por modo y forma

Se decidió dividir los resultados discriminando entre los distintos modos y formas verbales, en particular no se encontró ningún evento de modo imperativo en el corpus de testeo, la Tabla 41 muestra estos resultados.

Modo/forma	Indicativo	Subjuntivo	Infinitivo	Gerundio	Participio
CRF++	94,3%	44,4%	61%	50%	88,5%
Yamcha	95,45%	44,4%	63,4%	60%	96,3%

Tabla 41 - Resultados discriminados por modos y formas verbales

Modo/forma	Entrenamiento	Desarrollo	Testeo
Indicativo	928	253	184
Subjuntivo	56	17	9
Infinitivo	273	54	41
Gerundio	52	29	10
Participio	259	37	38
Imperativo	5	1	0

Tabla 42 - Cantidad de eventos de cada modo y forma en los distintos corpus.

Se observa que el corpus de testeo no tiene ningún verbo en imperativo, por esto no se tienen datos sobre su resultado. La forma verbal que más sorprende es el infinitivo, que siendo el segundo clasificado con mayor cantidad de eventos en el corpus de entrenamiento fue el tercero que se clasificó peor, siendo apenas superior a la forma verbal gerundio del que se contaba con un tercio de su cantidad en el corpus de entrenamiento. Esto hace pensar que el infinitivo es la forma verbal más complicada de clasificar, siendo su clasificación más complicada que la de los modos verbales también.

El corpus de testeo solo tiene 9 subjuntivos, de estos solo el 44,4% se clasificaron correctamente (4 verbos) pero se observa que el corpus de entrenamiento solo tiene 56 verbos en modo subjuntivo, por lo cual es razonable que este no se haya clasificado correctamente, dada la poca cantidad de datos. Lo mismo sucede con el gerundio, aunque este tuvo un mejor resultado. Lo opuesto pasa con el modo indicativo, del cual contamos con 928 eventos en el corpus de entrenamiento y este fue el que mejor se clasificó. Algo similar sucede con el participio, aunque solo contamos con 259 eventos en el corpus de entrenamiento.

8.5.3.2. Ejemplos de errores

Se muestran en esta sección algunos ejemplos clasificados incorrectamente por CRF++ pero no por Yamcha, o incorrectamente clasificados por Yamcha pero no por CRF++, o clasificados incorrectamente por ambos.

Ejemplos clasificados incorrectamente por CRF++ pero correctamente por Yamcha

La Tabla 43 muestra un evento realizado (*plantear*) clasificado incorrectamente por CRF++ como indefinido pero no por Yamcha. En cambio la Tabla 44 muestra un evento indefinido (*repitiera*) clasificado incorrectamente por CRF++ como realizado.

(8.1) *Esta vez, el ex director de la Oficina de Planeamiento y Presupuesto (OPP) y ex vicepresidente del Banco Central del Uruguay (BCU) opta por plantear su aspiración de manera directa y para eso se basa en los resultados de las últimas encuestas.*

Token	**Resultado**	**Yamcha**	**CRF++**
opta	realizado	realizado	realizado
por	O	O	O
plantear	**realizado**	**realizado**	**indef**
su	O	O	O
aspiración	O	O	O
de	O	O	O
manera	O	O	O
directa	O	O	O

Tabla 43 – Ejemplo de evento realizado clasificado incorrectamente por CRF++ pero no por Yamcha, parte del ejemplo mostrado en (8.1)

(8.2) *Cuando le preguntan si le gustaría que la escenita se repitiera con Bush, se va por los cerros de Úveda*.

Token	**Resultado**	**Yamcha**	**CRF++**
cuando	O	O	O
le	O	O	O
preguntan	realizado	realizado	realizado
si	O	O	O
le	O	O	O
gustaría	indef	indef	indef
que	O	O	O
la	O	O	O
escenita	O	O	O
se	O	O	O
repitiera	**indef**	**indef**	**realizado**
con	O	O	O
Bush	O	O	O
,	O	O	O

Tabla 44 – Evento indefinido clasificado incorrectamente por CRF++ pero no por Yamcha, parte del ejemplo mostrado en (8.2)

Ejemplos clasificados incorrectamente por Yamcha pero correctamente por CRF++

La Tabla 45 muestra un ejemplo de un evento (*tirar*) realizado clasificado incorrectamente como indefinido por Yamcha, pero correctamente por CRF++. La Tabla 46 muestra otro ejemplo de evento realizado (*tarde*) clasificado incorrectamente como indefinido por Yamcha, pero no por CRF++.

(8.3) *¿No se debería intentar convencer al votante del frente en vez de tirárselo en contra?*

Token	**Resultado**	**Yamcha**	**CRF++**
¿	O	O	O
No	O	O	O
Se	O	O	O
debería	indef	indef	indef
intentar	indef	indef	indef
convencer	indef	indef	indef
a	O	O	O
el	O	O	O
votante	O	O	O
de	O	O	O
el	O	O	O
frente	O	O	O
en_vez_de	O	O	O
tirar	**realizado**	**indef**	**realizado**
se	O	O	O
lo	O	O	O
en	O	O	O
contra	O	O	O
?	O	O	O

Tabla 45 - Evento realizado clasificado incorrectamente por Yamcha pero no CRF++, ejemplo (8.3)

(8.4) *Según Pujol, no puede ser que se tarde ocho meses para dar un permiso de residencia de trabajo a un ciudadano polaco, en un momento de falta de obra que actúa, en su opinión, como freno al crecimiento económico.*

Token	**Resultado**	**Yamcha**	**CRF++**
no	O	O	O
puede	O	O	O
ser	no_realizado	no_realizado	no_realizado
que	O	O	O
se	O	O	O
tarde	**realizado**	**indef**	**realizado**
ocho	O	O	O
meses	O	O	O

Tabla 46 – Evento realizado clasificado incorrectamente por Yamcha pero no por CRF++, parte del ejemplo mostrado en (8.4)

Ejemplos clasificados incorrectamente por ambos toolkits

La Tabla 47 muestra dos eventos realizados (*viene* y *demostrar*) clasificados incorrectamente como indefinidos tanto por Yamcha como por CRF++. A su vez la Tabla 48 muestra un ejemplo de un evento (*sumen*) indefinido clasificado incorrectamente como realizado tanto por Yamcha como por CRF++.

(8.5) *En este caso, la Encuesta Nacional Factum del otro día viene a demostrar que se confirma una tendencia.*

Token	**Resultado**	**Yamcha**	**CRF++**
la	O	O	O
Encuesta_Nacional_Factum	O	O	O
de	O	O	O
el	O	O	O
otro	O	O	O
día	O	O	O
viene	**realizado**	**indef**	**indef**
a	O	O	O
demostrar	**realizado**	**indef**	**indef**
que	O	O	O
se	O	O	O
confirma	realizado	realizado	realizado
una	O	O	O
tendencia	O	O	O

Tabla 47 – Eventos realizados clasificado incorrectamente por Yamcha y CRF++, parte del ejemplo mostrado en (8.5)

(8.6) *En una carta que difundió públicamente este fin de semana, el candidato a la Intendencia Municipal de Montevideo (IMM) por Alianza Nacional (AN), Javier De Haedo convocó a los votantes del Partido Colorado (PC) a que se sumen a su proyecto.*

Token	**Resultado**	**Yamcha**	**CRF++**
Javier_De_Haedo	O	O	O
convocó	realizado	realizado	realizado
a	O	O	O
los	O	O	O
votantes	O	O	O
del	O	O	O
partido_colorado	O	O	O
(	O	O	O
PC	O	O	O
)	O	O	O
a	O	O	O
que	O	O	O
se	O	O	O
sumen	**indef**	**realizado**	**realizado**
a	O	O	O
su	O	O	O
proyecto	O	O	O

Tabla 48 - Evento indefinido clasificado incorrectamente por Yamcha y CRF++, parte del ejemplo mostrado en (8.6)

Capítulo 9

Conclusiones y Trabajos Futuros

9.1. Conclusiones

El fin del proyecto era crear una herramienta que determine la factividad de cada evento en el texto. Para ello fue necesario generar un modelo de aprendizaje automático que tome en cuenta los distintos atributos que se encuentran en el contexto de los eventos y determine, basado en ellos, su factividad.

Esta tarea se realizó con éxito utilizando dos *toolkits* (CRF++ y Yamcha) los cuales obtuvieron una Medida F de 85,1% y 87,4% respectivamente, superando ampliamente la línea base (68,5%), aunque teniendo aún espacio para mejorar ya que no se alcanzó la línea tope (90,4%).
La línea base consistió en un sistema simple basado en reglas que determina la factividad de los eventos en el texto. La línea tope, por otro lado, muestra la concordancia que existe entre las dos estudiantes de lingüística que anotaron manualmente el corpus. Se observa que CRF++ supera la línea base por 16,6% y Yamcha lo hace por 18,9%. Estos resultados superan los reportados por [RS08] (reporta una Medida F entre 74% y 85%) y por [SP12] (reporta una Medida F entre 70% y 80%) lo cual hace pensar que los resultados son aceptables, siendo el primer trabajo que determina la factividad para el idioma Español. Se observa que estos trabajos están basados en reglas y no en métodos de aprendizaje automático, pero no contamos con datos de trabajos de este tipo.

Por último coincidimos con las conclusiones obtenidas en [GSB12] que indican que determinar la certeza de los eventos presenta resultados satisfactorios (en nuestro caso los eventos realizados y no realizados), mientras que la posibilidad no obtiene resultados tan satisfactorios (en nuestro caso los indefinidos).

Teniendo en cuenta que este sistema utiliza otras dos herramientas, cada una con una exactitud determinada, se deben considerar dichos valores de exactitud para medir realmente qué tan exacto es nuestro sistema. Freeling tiene una exactitud de 97% para obtener la categoría y modo y 95% en el POSTAG completo. El analizador de dependencias de MaltParser tiene una exactitud de entre 80% y 90%. Teniendo esto en cuenta, un 85% de exactitud en nuestro sistema parece bastante razonable considerando que MaltParser solo tiene una exactitud que ronda por los mismos valores. Estos valores de exactitud muestran qué tan complejo es alcanzar la línea tope de 90,4% sabiendo que las herramientas utilizadas no alcanzan dicha exactitud, lo que a su vez demuestra lo complejo que es el problema a resolver con las herramientas existentes para el idioma español.

Los aportes a nuestra formación fueron varios. Se aprendieron conceptos básicos de lingüística y se profundizó en el campo de aprendizaje automático, especialmente con los modelos CRF y SVM. También se profundizó en la recolección de atributos, experimentación y testeo. A su vez, se aprendió a trabajar en un proyecto extenso, siguiendo un procedimiento establecido, donde primero se hizo un estudio del estado del arte, luego se ideó la solución, para luego implementarla y obtener los resultados. Por último se aprendió a utilizar el lenguaje Python.

9.2. Trabajos Futuros

Considerando que nuestro marco de trabajo TEMANTEX es el mismo que el proyecto de grado de reconocimiento de eventos tiene sentido realizar un módulo de integración para dicho proyecto con el fin de generar una herramienta más completa para la lengua española para el mundo PLN que sea capaz de etiquetar los eventos con sus factividades. Una vez integrado dicho módulo se puede utilizar sobre un conjunto nuevo de textos en español para brindar un corpus pre procesado (etiquetado con eventos y factividades) y así facilitar la tarea de futuros anotadores. De esta forma, estos ya no tendrían que etiquetar un conjunto de textos desde cero sino que deberían corregir el corpus pre procesado, obteniendo como resultado en menor tiempo y con menos esfuerzo, un corpus de mayor tamaño que retroalimente a la herramienta. Esto fue lo realizado con el segundo corpus del proyecto, aunque solo para la factividad.

Al tener un corpus lo suficientemente grande empieza a ser factible extender los valores del atributo factividad, en particular desagrupar el actual atributo indef por los cuatro atributos originales (posible, futuro programado, futuro negado e indefinido) con esto se puede modelar mejor la determinación de la factividad de los eventos mencionados en el texto.

También se podría plantear la posibilidad de armar un módulo basado en [AR11] con el fin de realizar un modelado más completo sobre las factividades de los eventos incluyendo las fuentes del texto y la factividad relativa a dichas fuentes. La creación de este módulo también abre una gran variedad de atributos para el aprendizaje de nuestro proyecto.

Finalmente, se podría integrar nuestro sistema a un sistema de recuperación de información, o de respuestas automáticas a preguntas para así mejorar sus resultados. Ya que como se mencionó en la sección 1.2 estos sistemas requieren de la determinación de la factividad para mejorar su desempeño.

Bibliografía

[PCI+03]	James Pustejovsky, José Castaño, Robert Ingria, Roser Saurí, Robert Gaizauskas, Andrea Setzer, y Graham Katz. TimeML: Robust specification of event and temporal expressions in text. In Fifth International Workshop on Computational Semantics, 2003.
[LRH89]	L. R. Horn. A Natural History of Negation. University of Chicago Press, Chicago, 1989.
[WMR08]	Dina Wonsever, Marisa Malcuori, y Aiala Rosá. SIBILA: Esquema de anotación de eventos. Reporte Técnico RT 08-11, PEDECIBA, 2008.
[WMR09]	Dina Wonsever, Marisa Malcuori, y Aiala Rosá. Factividad de los eventos referidos en textos. Reporte Técnico RT 09-12, PEDECIBA, 2009.
[AR11]	Aiala Rosá. Identificación de opiniones de diferentes fuentes en textos en español. Tesis de Doctorado, PEDECIBA, 2011.
[SP07]	Roser Saurí, James Pustejovsky. Determining Modality and Factuality for Text Entailment. International Conference on Semantic Computing, 2007.
[NCK06]	Rowan Nairn, Cleo Condoravdi, Lauri Karttunen. Computing relative polarity for textual inference. Proceedings of the Fifth International Workshop on Inference in Computational Semantics, 2006.
[SVP06a]	Roser Saurí, Marc Verhagen, James Pustejovsky. SlinkET: A Partial Modal Parser for Events. In Language Resource and Evaluation Conference, 2006.
[FRE12]	Xavier Carreras, Isaac Chao, Lluís Padró, Muntsa Padró. Freeling: An Open Source Suite of Language Analyzers, Universitat Politecnica de Catalunya, 2004.
[FREELING]	http://nlp.lsi.upc.edu/freeling/ Última visita: 16/03/2014
[M1.7.2]	http://www.maltparser.org/ Última visita: 16/03/2014
[LCE]	http://lse.umiacs.umd.edu/ Última visita: 16/03/2014
[GSE]	http://www.google.com/coop/cse Última visita: 16/03/2014
[BNC]	http://www.natcorp.ox.ac.uk/ Última visita: 16/03/2014
[ANC]	http://americannationalcorpus.org Última visita: 16/03/2014
[EAGLES]	http://nlp.lsi.upc.edu/freeling/doc/tagsets/tagset-es.html Última visita: 16/03/2014
[CONLL]	http://www.conll.org/ Última visita: 16/03/2014

[CONLL-X] Parsing Sabine Buchholz, Erwin Marsi. CoNLL-X shared task on Multilingual Dependency. CoNLL-X '06 Proceedings of the Tenth Conference on Computational Natural Language Learning, 2006.

[CRF++] http://crfpp.googlecode.com/svn/trunk/doc/index.html
Última visita: 16/03/2014

[SVP06b] Roser Saurí, Marc Verhagen, James Pustejovsky. Annotating and Recognizing Event Modality in Text. In the 19th International FLAIRS Conference, FLAIRS 2006.

[AD10] Alan Descoins. Reconocimiento automático de eventos en textos. Proyecto de Grado Facultad de Ingeniería, 2010.

[DES11] Alan Descoins. Métodos de Clasificación Secuencial, 2011.

[TIM08] TimeML Working Group. TimeML 1.2.1. A Formal Specification Language for Events and Temporal Expressions. Octubre 2005.

[TIMEML] http://www.timeml.org/site/timebank/timebank.html
Última visita: 16/03/2014

[TIMEBANK] http://www.timeml.org/site/timebank/documentation-1.2.html
Última visita: 13/04/2014

[SAU08] Roser Saurí, Jessica Littman, Bob Knippen, Robert Gaizauskas, Andrea Setzer y James Pustejovsky. TimeML Annotation Guidelines Version 1.2.1. Enero 2006.

[TAR08] Proyecto Tarsqi, http://www.timeml.org/site/tarsqi/index.html
Última visita: 16/03/2014

[KAR70] Lauri Karttunen. Some observations on factivity. Papers in Linguistics, volume 4, issue 1, pages 55-69, 1971.

[SKVP05] Roser Saurí, Robert Knippen, Marc Verhagen, James Pustejovsky. Evita: A Robust Event Recognizer For QA Systems. HLT '05 Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing, 2005.

[GSB12] Goran Glavaš, JanŠnajder, Bojana Dalbelo Bašíc. Are you for real? Learning event factuality in Croatian texts. Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD 2012). Ljubljana, Slovenia, October 2012.

[RS08] Roser Saurí A Factuality Profiler for Eventualities in Text. Ph.D. Thesis. Brandeis University, 2008.

[SP12] Roser Saurí, James Pustejovsky. Are You Sure That This Happened? Assessing the Factuality Degree of Events in Text. Computational Linguistic, Volume 38 Issue 2, June 2012, pages 261-299.

[LP11] Lluís Padró. Analizadores Multilingües en FreeLing. "Linguamática", Diciembre 2011, vol. 3, núm. 1, p. 13-20.

[LEO99] Leonetti, Manuel. “El artículo”. En: I. Bosque y V. Demonte (dirs.). Gramática Descriptiva de la Lengua Española. Madrid: Espasa, 1999. Vol. 1, p. 787-890.

[KIP70] Kiparsky P., Kiparsky. Fact. Manfred Bierwisch and Karl Heidolph Eds, Progress in Linguistics. A Collection of Papers, pages 143-173. Mouton, The Hague, Paris, 1970.

[H75] Hooper, J. B. On assertive predicates. In J. Kimball (Ed.), Syntax and semantics, IV (pp. 91–124). New York, USA: Academic Press, 1975.

[LMCP01] John Lafferty, Andrew McCallum, Fernando Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. ICML'01 Proceedings of the Eighteenth International Conference on Machine Learning, Pages 282-289, 2001.

[MB07] Ben Medlock, Ted Briscoe. Weakly Supervised Learning for Hedge Classification in Scientific Literature. In Proceedings of ACL-07, the 45th Annual Meeting of the Association of Computational Linguistics, 2007.

[Guy03] Isabelle Guyon, André Elisseeff. An Introduction to Variable and Feature Selection. Journal of Machine Learning Research 3 (2003) 1157-1182.

[MW13] Marisa Malcuori, Dina Wonsever. Lista de verbos implicativos y fácticos, 2013. Manuscrito no publicado.

[KA11] http://www.eps.uam.es/esp/alumnos/trabajos_fin_master/Khayyat_Arranz_Sergio-Nabil.pdf
Última visita: 16/03/2014

[DFCoNLL] http://nextens.uvt.nl/depparse-wiki/DataFormat
Última visita: 16/03/2014

[CoNLL07] The CoNLL 2007 Shared Task on Dependency Parsing. Joakim Nivre, Johan Hall, Sandra Kubler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, Deniz Yure

Anexo

A. Resúmenes de trabajos leídos

A Factuality Profiler for Eventualities in Text, Roser Saurí i Colomer (2008)

El objetivo de este trabajo es diseñar y desarrollar un **perfilador de factividad**, es decir, una herramienta que determine la factividad de todos los eventos del texto.

Se mencionan dos propiedades que este perfilador de factividad debe cumplir:

1. Debe basarse únicamente en expresiones lingüísticas que señalen la factividad.
2. Cuando corresponda, debe ser capaz de representar información diferente (y posiblemente contradictoria) sobre la factividad de un evento.

La factividad es definida en este trabajo como "la categoría a cargo del estado factual de los eventos". Es decir, marcan si un evento es presentado como fáctico (sucedió), contra fáctico (no sucedió) o si su ocurrencia o no ocurrencia es incierta.

La factividad de un evento puede ir desde verdad factual a contra factual, pasando por una amplia gama de modalidades, entre ellos:

1. Grados de posibilidad: *Pero al mismo tiempo, el fallo dijo que no se probó que hubiera contaminación en el Río Uruguay y por lo tanto UPM **puede** seguir funcionando en su actual emplazamiento.*

2. Creencia: *Algunos analistas chinos **creen** que los Estados Unidos continuarán provocando a Corea del Norte.*

3. Evidencialidad: *El subcomandante Marcos **dijo** que el gobierno mexicano no está interesado en poner fin al conflicto.*

4. Expectativa: *Hans Blix **quiere** que los EE.UU. permitan que los inspectores de la ONU vuelvan a Irak para verificar las armas encontradas por las fuerzas de la coalición.*

5. Intento: *George Mallory y Andrew Irvine **intentaron** escalar el Everest por primera vez en 1924.*

6. Orden: *John Murtha **pidió** la retirada inmediata de las tropas estadounidenses de Irak.*

La factividad es representada por una escala de doble eje, siendo un eje el de la **modalidad epistémica** (el grado de compromiso del autor sobre la certeza de lo que se afirma, que va desde incierto hasta absolutamente seguro), y el de la polaridad (que consiste de solo dos valores: positiva y negativa), representándola entonces como un par <modalidad, polaridad>. Se define entonces a la factividad como muestra la Tabla 49.

Modalidad\Polaridad	Positivo (+)	Negativo (-)	Desconocido
Cierto	Fáctico <CT,+>	Contra fáctico <CT,->	Cierto pero de salida desconocida <CT,UN>
Probable	Probable <PR,+>	No probable <PR,->	(NA)
Posible	Posible <PS,+>	No posible <PS,->	(NA)
Desconocido	(NA)	(NA)	Desconocido o no comprometido <UN,UN>

Tabla 49 - Valores de Factividad

Partículas de Polaridad

Determinan el eje de polaridad de la factividad de un evento, es decir, si este es positivo o negativo. Por defecto la polaridad es positiva a menos que exista algún marcador de polaridad. A no ser que estos elementos se encuentren combinados con otros, como ser las partículas de modalidad, no dejan lugar a la incertidumbre.

Partículas de Modalidad

Dado un evento, las partículas de modalidad le dan distintas tonalidades a su factividad. A diferencia de las partículas de polaridad, pueden dejar lugar a la incertidumbre y se basan en una escala no discreta.

Hay dos fuentes principales según el nivel donde se encuentren dentro del contexto del evento: (A.1) auxiliares modales, que se encuentran en un nivel local (elementos expresados verbalmente), y (A.2) adverbios modales clausales y oracionales, que se encuentran en un nivel no local.

(A.1) a. Posibilidad: *podría*, *puede*.
b. Obligación: *debe*, *tiene que*.
c. Necesidad: *necesitar*.

(A.2) a. Posibilidad: *probablemente*, *tal vez*.
b. Frecuencia: *normalmente*, *siempre*.

Contextos de subordinación basados en léxico

Los SSP (predicados de selección de situaciones) son “los predicados que seleccionan un argumento que denotan un evento o situación”. En muchos casos estos ayudan a determinar la factividad del evento.

Los SSP pueden expresar, entre otros:

- Diferentes grados de certeza/posibilidad sobre la situación que indican. Algunos ejemplos son los predicados asertivos débiles [H75] (*pensar*, *creer*, *suponer*) y algunos predicados asertivos fuertes (*afirmar*, *decir*).
- Factividad de la situación que indican (*lamentar*, *olvidar)*.
- Evidencialidad: *decir*, *reportar*.
- Orden: *obligar*, *ordenar*.
- Compromiso: *ofrecer*, *proponer*.
- Voluntad: *querer*, *desear*.
- Acuerdo/plan: *decidir*, *acordar*, *planear*.
- Posibilidad: *es posible que*.
- Inclinación: *dispuesto*, *listo*, *entusiasmado*, *reacio*.

Contextos de subordinación basados en construcciones

Algunas relaciones de subordinación son provocadas por una relación de diálogo entre dos cláusulas, éstas introducen información de factividad de algún tipo. Pueden dividirse, entre otros, en (los eventos se muestran subrayados):

- **Cláusulas relativas:** presupone al evento como verdadero. Ejemplo:
  (A.3) *Rice, quien se convirtió en secretario de estado hace dos meses, ha hecho un balance de un período de cambios tumultuosos.*

- **Estructuras *cleft*:** el evento se presupone como verdadero. Ejemplo:
  (A.4) *Fue el señor Bryant quien, el 19 de julio de 2001, solicitó al representante Bartlett que escriba y entregue una carta dirigida a él.*

- **Cláusulas de propósito:** el evento denotado es intencional de naturaleza. Ejemplo:
  (A.5) *La comisión ambiental debe adoptar reglamentos para asegurar que las personas no se vean expuestas a los residuos radiactivos.*

- **Construcciones condicionales:** el evento denotado es intencional y depende de la factividad del evento denotado en la cláusula antecedente (en negrita) que también es intencional. Ejemplo:
  (A.6) *El 2 de diciembre Marcos se comprometió a volver a la mesa de negociaciones si se* ***desmilitariza*** *la zona de conflicto.*

- **Cláusulas subordinadas temporales:** el evento se presupone como verdadero. Ejemplo:
  (A.7) *Mientras Chomsky estaba revolucionando la lingüística, el resto de las ciencias sociales se había dormido.*

Fuentes de Factividad

Cuando se menciona un evento, existe una fuente que se compromete con la factividad de ese evento. Por esto, la factividad de los eventos es relativa a los participantes, referidos en este trabajo como **fuentes de factividad**.

Se define fuente como "los individuos cognitivos que asignan valores de factividad a los eventos mencionados en el texto".

Las fuentes son introducidas en el texto mediante predicados epistémicos, entre los cuales se encuentran los predicados de reporte (*decir*), de conocimiento y opinión (*creer, saber*), reacciones psicológicas (*lamentar*), etc.

Distintos participantes pueden tener distintas perspectivas sobre el valor de factividad del mismo evento. Esto se observa con el siguiente ejemplo, donde el autor (A.9), Juan (A.10) y María (A.11) tienen distintas perspectivas del evento *Juan está enfermo* en (A.8).

(A.8) *María lamenta que Juan no sepa que está enfermo.*
(A.9) Juan está enfermo. Verdadero
(A.10) Juan sabe que está enfermo. Falso
(A.11) María sabe que Juan está enfermo. Verdadero.

El autor de la oración es referido como **indicador**, y la fuente que menciona el evento como **conocedor**. Esto se debe a que el **indicador** es quien presenta el compromiso del **conocedor** sobre dicho evento.

Perfilador de Factividad

Debido a lo mencionado anteriormente, en la factividad participan:

1. El evento en foco, e.
2. La fuente que asigna un valor de factividad a ese evento e.
3. El valor de factividad siendo asignado.
4. El momento de asignación de ese valor de factividad, t.

El perfilador de factividad se define como el conjunto de valores de factividad que las diferentes fuentes asignan a un evento e: $f_p(e) = \{< s, t, f >\}$
Donde s es una fuente, t es una expresión temporal y f es un valor de factividad de la forma <mod, pol>.

El perfilador de factividad es un decodificador ingenuo, por lo que cada oración se inicia con el valor <CT,+> y este valor puede luego ser cambiado por los marcadores de factividad disponibles en el texto. Estas modificaciones se realizan acorde a las Tablas 50 y 51, donde la primera modifica la polaridad y la segunda la modalidad.

Valor del marcador	Polaridad del Contexto		
	+	**-**	**UN**
+	+	-	UN
-	-	+	UN

Tabla 50 - Valor de polaridad dada la polaridad del contexto

Marcador	Factividad del Contexto											
	Polaridad = +				Polaridad = -				Polaridad = UN			
	CT	**PR**	**PS**	**UN**	**CT**	**PR**	**PS**	**UN**	**CT**	**PR**	**PS**	**UN**
CT	CT	PR	PS	UN	PS	PR	PS	UN	CT	PR	PS	UN
PR	PR	PR	PS	UN	PR	PR	PS	UN	PR	PR	PS	UN
PS	PS	PS	PS	UN	CT	PR	PS	UN	PS	PS	PS	UN
UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN	UN

Tabla 51 - Valor de modalidad dada la factividad del contexto

Learning event factuality in Croatian texts, Goran Glavaš, JanŠnajder, Bojana Dalbelo Bašíc (2012)

Su objetivo es identificar la certeza (nivel de confianza sobre la realización del evento) y polaridad (ocurrencia o no ocurrencia) de los eventos, así como determinar la viabilidad de este tipo de herramientas para las lenguas con pocos recursos.

Sus resultados indican que si bien utilizar características basadas en el léxico para la polaridad produce buenos resultados, la predicción de la certeza requiero del uso de características más sofisticadas. Sin embargo, sus estadísticas dicen que estos resultados no son significativos comparados con una línea base simple basada en reglas.

El módulo define la factividad como el par <certeza, polaridad> donde la certeza puede tomar los valores cierto, probable y posible, y la polaridad puede tomar los valores positivo y negativo. Las características del léxico que se utilizan para realizar el aprendizaje se detallan a continuación, estas determinan tanto la clasificación de la polaridad como la clasificación de la certeza:

1. *token*, lema y lexema del evento.

2. Final del evento, es el sufijo de la última vocal del *token* o la penúltima vocal en caso de terminar en vocal.

3. Descripción morfosintáctica del evento, dicho valor se adquiere semiautomáticamente.

4. Conjunto de *tokens* del contexto izquierdo y derecho del evento, se define previamente el contexto como una ventana de cinco *tokens*. Se consideran dos conjuntos de *tokens*, el primero contiene el contexto izquierdo y el segundo el contexto derecho.

5. Lemas de los primeros *tokens* anteriores y posteriores al evento.

6. Tipo de evento: el tipo de evento basado en TimeML ([TIM08]):
    a. Ocurrencia: *morir*, *chocar*, *construir*, *unir*, *vender*.
    b. Estado: *a bordo*, *secuestrado*, *amor*.
    c. Reporte: *decir*, *informar*, *anunciar*.
    d. I-Action: *intento*, *intentar*, *promesa*, *oferta*.
    e. I-State: *creer*, *tener intención*, *querer*.
    f. Aspectual: *comenzar*, *terminar*, *parar*, *continuar*.
    g. Percepción: *ver*, *oír*, *mirar*, *sentir*.

7. Sustantivo verbal y deverbal, una característica binaria que indica si el lexema del evento es un sustantivo verbal o deverbal, ya que se observa que los eventos expresados tanto con sustantivos verbales como deverbales tienden a ser más hipotéticos.

8. Oración interrogativa, una característica binaria que indica si la frase que contiene el evento es interrogativa. Este tipo de evento tiende a ser incierto, ya que en la pregunta hay algo que no se conoce.

9. Argumento de otro evento, por lo general los eventos que son argumentos directos de otro evento son más propensos a ser no factuales, con lo cual se tiene una característica binaria que indica si el evento es un argumento de otro evento.

A continuación se detallan las características que competen únicamente a la clasificación de polaridad:

1. Pistas negativas encontradas en el contexto izquierdo, se utilizó una lista de pistas negativas más frecuentes, donde las mismas son conjugaciones de *no ser* y *no querer*, y, además las palabras, *no*, *nadie*, *nunca*, *tampoco*. Estas palabras son buscadas en el contexto del lado izquierdo del evento, el cual se define como todos los *tokens* de la frase anteriores al evento.

2. Pistas negativas encontradas en el contexto izquierdo inmediato, al igual que en la parte anterior se utiliza la lista de pistas negativas, pero en este caso se toma un contexto izquierdo de tres *tokens* con respecto al evento. Al considerar el contexto inmediato y el lejano, se busca reconocer qué tanto influyen ambos modificadores de polaridad en el evento.

3. Distancia entre el evento y la pista más cercana negativa.

Para terminar se especifican las características que corresponden a la clasificación de certeza:

1. Pistas condicionales encontradas en el contexto izquierdo, se utilizó una lista de pistas condicionales más frecuentes, donde las mismas son la palabra *si*, y las formas flexivas del verbo *querer*. Estas palabras son buscadas en el contexto izquierdo del evento. Las pistas condicionales indican fuertemente si un evento es hipotético, con lo cual, pueden ser muy importantes para predecir la categoría de certeza en cual se ubica.

2. Pistas condicionales encontradas en el contexto izquierdo y derecho, se utiliza un conjunto binario de atributos, uno para cada contexto (izquierdo, derecho) del evento. Para este caso se buscan las mismas pistas condicionales definidas en el atributo anterior pero en un entorno de tres *tokens* con respecto al evento.

3. Distancia entre el evento y la pista condicional más cercana.

4. Pistas en tiempo futuro encontradas en el contexto izquierdo. Dado que los acontecimientos a futuro aún no se han producido, estos generan una mayor incertidumbre. Para esto se armó una lista de pistas que se utilizan para expresar el tiempo futuro, conjugaciones del verbo *querer* y las formas en tiempo futuro perfecto del verbo *ser*. Las pistas se buscan en el contexto izquierdo del evento.

5. Pistas en tiempo futuro que se encuentran en el contexto inmediato izquierdo y derecho: utilizando la misma lista que el caso anterior se arman dos atributos, uno por cada contexto, en un entorno de tres *tokens* con respecto al evento

6. Distancia entre el evento y la pista en tiempo futuro más cercana.

7. Pistas de posibilidad encontradas en el contexto izquierdo, utiliza una lista de pistas donde por definición estas se relacionan fuertemente con la incertidumbre y la posibilidad. Esta lista está formada por conjugación de *poder*, además de las palabras *tal vez* y *posible*. Las pistas se buscan en el contexto izquierdo del evento.

8. Pistas de posibilidad que se encuentran en el contexto inmediato izquierdo y derecho, utilizando las mismas pistas que en el caso anterior, se arman dos conjuntos de características, uno por cada contexto, en un entorno de tres *tokens* con respecto al evento.

9. Distancia entre el evento y la pista de posibilidad más cercana.

El aprendizaje sobre la clasificación de la polaridad y certeza se realiza utilizando *Support Vector Machine* (SVM). El corpus está formado por la extracción de eventos y relaciones temporales en noventa documentos del periódico Vjesnik realizado por cinco anotadores. Además sobre el conjunto anterior dos anotadores realizaron la anotación de la polaridad y certeza (cada uno anotó una mitad) quedando como resultado un conjunto de 4596 eventos. Un 78,6% de los eventos fue anotado como positivo y seguro, algo esperado en el género del periódico.

Identificación de opiniones de diferentes fuentes en textos en español (Aiala Rosá)

Esta tesis abarca el estudio de expresiones que transmiten opiniones de diferentes fuentes. La tesis contiene un modelo para los predicados de opinión y sus argumentos (fuente, asunto y mensaje). También se creó un léxico de predicados de opinión con información del modelo. Se implementaron tres sistemas informáticos estos son (se muestra para cada uno la Medida F sólo para la determinación de las fuentes):

- El primero está basado en reglas contextuales y reporta una Medida F de 79%.
- El segundo fue desarrollado basado en el modelo CRF y tiene una Medida F de 76%.
- El tercero es una combinación de los dos anteriores y consiste en agregar la salida del primero como nuevo atributo en el segundo sistema. Este sistema reporta el mejor resultado de los tres, teniendo una Medida F de 83%.

Tomando como referencia la Medida F de otros idiomas (valor entre 63% y 89,5%) el valor hallado de 83% es muy satisfactorio.

En el transcurso de este proyecto se generaron diversos recursos para el procesamiento automático: se creó un léxico de predicados de opinión, se creó un corpus de 13.000 palabras anotadas con las opiniones y sus elementos, y también un corpus con 40.000 palabras anotadas con los predicados de opinión y sus fuentes.

La opinión es un fragmento de texto que transmite lo que alguien opina o dice sobre algo, consta de varios argumentos:

- **Fuente:** Autor de la opinión.
- **Asunto:** Tema sobre el que se opina.
- **Mensaje:** Contenido de la opinión.
- **Predicado de opinión:** Por lo general es un verbo de comunicación o clases semánticas subjetivas, pero también puede ser un nombre o una preposición o locución prepositiva.
- **Orientación semántica:** Propiedad semántica para la opinión, puede tomar el valor positivo, neutro o negativo.

Este trabajo tiene como objetivo elaborar una definición de opinión y establecer los elementos que la componen, además de definir un modelo que contemple las propiedades sintácticas y semánticas de las expresiones que transmiten opinión. También tiene el objetivo de desarrollar un sistema informático para el reconocimiento automático de las opiniones, y junto con este objetivo se encuentra el de investigar la interacción entre métodos simbólicos y métodos estadísticos.

El sistema de reglas contextuales, que utiliza un intérprete de Prolog, aborda los siguientes aspectos:

- La identificación de predicados de opinión verbal, nominal y preposicional.
- La identificación de las fuentes de opinión, siempre que este se mencione explícitamente dentro de la cláusula correspondiente a un predicado identificado.

- La identificación del asunto en los casos en los que este es mencionado en un constituyente específico:
    - Grupos preposicionales introducidos por *sobre*, *respecto a*, *en lo referente a* y expresiones similares.
    - Grupos nominales que son objetos directos de verbos como *apoyar*, *rechazar*, etc.
    - Complementos del nombre para casos de predicados nominales, como el rechazo hacia Y.
- La identificación del mensaje en sus diferentes formas:
    - Cita directa entre comillas, cinta indirecta, expresiones mixtas con fragmentos entre comillas dentro de citas indirectas, entre otros.
- Resuelve la determinación de la orientación semántica del predicado y del asunto, ya que se trata de un valor asociado a las piezas léxicas. También para algunos casos resuelve la fuente.

Se destaca que, además de los sistemas para el reconocimiento de la opinión y sus elementos, se generó un conjunto de recursos importantes para el español, un repertorio de predicados de opinión con información sintáctico-semántica asociada, un corpus de 13.000 palabras anotado con las opiniones y sus elementos (predicado, fuente, asunto y mensaje) y un corpus de 40.000 palabras anotado con predicados de opinión y fuentes.

El estudio teórico realizado y las herramientas y recursos desarrollados constituyen un importante aporte para el desarrollo del Procesamiento de Lenguaje Natural aplicado a textos en español, en particular, para el área Minería de Opiniones en donde hay muy pocos trabajos orientados a este idioma.

B. Léxicos especializados

Predicados implicativos

La siguiente tabla muestra los distintos verbos que tienen alguna implicancia sobre el evento dependiendo de la polaridad del mismo. La primera columna muestra el verbo que genera implicancia y la segunda la polaridad resultativa según dicho verbo tenga polaridad positiva o negativa.

Verbo	**Polaridad resultativa**	
	+	**-**
logra	+	-
olvidar(se) de + inf	-	+
verse obligado a	+	
negarse	-	
intentar		-
dudar en/de + inf		+
olvidar(se) que	+	+
fingir	-	-
simular	-	-
hacer creer	-	-
querer		
acordar	+	+
adivinar	+	+
advertir	+	+
aprender	+	
asimilar	+	
averiguar	+	
cerciorar	+	
colegir	+	
comprender	+	+
comprobar	+	
concluir	+	
confirmar	+	
constatar	+	
cuestionar	+	+
demostrar	+	
descartar	-	
desconocer	+	+
documentar	+	
enterarse	+	+
estipular	+	
evidenciar	+	
ignorar	+	+
imaginar		+
olvidar	+	+

probar	+	
recordar	+	+
resolver	+	
saber	+	+
soñar		+
sopesar	+	+
verificar	+	
visualizar	+	+
convencer	+	-
creer		+
disuadir	-	+
dudar		+
esperar		+
estimar		+
evocar	+	+
fraguar	-	
imaginar		+
intuir		+
elucubrar	-	
persuadir	+	
presentar	+	+
presumir		+
prever	+	+
sospechar		+
suponer		+
titubear		+
vacilar		+
perdonar	+	+
justificar	+	+
condenar	+	+
aprobar	+	+
criticar	+	+
reconocer	+	+
dejar	+	-
frustrar	-	+
impedir	-	+
imposibilitar	-	+
permitir	+	-
tolerar	+	+
acostumbrar	+	
aficionar	+	
ayudar	+	
favorecer	+	
habituar	+	
impeler	+	
impulsar	+	
incitar	+	
inclinar	+	
inducir	+	
promover	+	

obstruir	-	+
forzar	+	
imponer	+	
obligar	+	
poner	+	-
intentar		-
tratar		-
pretender	-	+
atrever	+	-
evitar	-	+
disponer	+	-
procurar		-
negar	-	
empeñar	+	
apresurar	+	
proceder	+	-
acertar	+	-
arriesgar	+	-
rehuir	-	+

Palabras que indican Negación, Posibilidad e Implicaciones

Palabras que indican negación

Lema	**Ejemplo**
negar	Juan negó haber copiado en el examen.
rechazar	Rechazaron apelación por la causa de Leandro Centeno.
evitar	Se evitó que explotara la granada en Iraq.
fallar	Microsoft falló en su intento de resucitar la marca Nokia
faltar	A Germán solo le falta ordenar su cuarto.
falso	Es falso que las Google Glass salgan hoy a la venta.
ignorar	Uruguayos ignoran que son espiados por el gobierno.
no	Andrés Calamaro no apareció en el escenario.
nunca	Aunque nunca escuche su disco, iré a su concierto.
nada	Nada le hace reír.
sin	Debido a una enfermedad pasó varios días sin ir a clase.
ninguno	Este año ningún día cayó agua nieve.
nadie	Nadie quiere a Martin.
tampoco	Virginia tampoco abrió la puerta.
jamás	En Uruguay jamás nieva en verano.

Palabras que indican posibilidad

Lema	**Ejemplo**
parecer	Al parecer la ventana está abierta.
procurar	María procuró lavar los pisos.
intentar	El intentó cantar a capela.
prometer	Andrés prometió lavar los platos.
probar	Matías decidió probar levantarse temprano.
intento	Javier hizo un intento de mirar la película.
promesa	Sofía le hizo la promesa de cuidar a sus plantas.
conjeturar	El detective conjeturó que Manolo había robado el diamante.
estimar	Para el año próximo Leonardo Di Caprio estima que ganará un Oscar.
pronosticar	Para mañana se pronosticó que sería un día lluvioso.
especular	Julio no hace más que especular sobre lo que haría si ganara la lotería.
hipótesis	María tiene la hipótesis de que mañana va a llover.
presunción	El gerente tiene la presunción de que el producto será un éxito.
suposición	El canal del clima hizo la suposición de que el lunes va a estar nublado.
sospecha	Tengo la sospecha de que alguien entró al cuarto.
acusar	Fue acusado de robar un banco.
presunto	Twitter mejora la seguridad tras el presunto espionaje de la NSA.
rumor	Los rumores apuntan a que Lucía y Marcelo están trabajando en el proyecto.
tratar	Juan trató de salvar el examen.
preguntar	Juan se preguntó si afuera llueve.
suponer	Supongo que a esta altura ya debe haber llegado el avión.
sospechar	La NASA sospecha que hay vida en Marte.
considerar	Los profesores consideran suspender las vacaciones de Setiembre.
pedir	Laura pidió que recorrijan su prueba.
desconocer	Se desconoce si el embajador ya partió hacia Haití.
dudar	Sara duda que salve el examen de manejo.
dudoso	En época de sequía es dudoso que llueva.
implicar	Para él las vacaciones implican descansar.
imposible	Es imposible que en verano caiga nieve.
imposibilidad	La investigación se cerrará por la imposibilidad de conseguir financiación.
improbabilidad	La improbabilidad de que venga es muy alta.
improbable	Es improbable que en el desierto llueva.
percibir	Andrés percibe que alguien lo está siguiendo.
posibilidad	Está la posibilidad de que llueva mañana.
capaz	Capaz mañana vienen unos amigos a casa.
posible	Es posible que mañana se mejore de la gripe.
oportunidad	El Real Madrid tiene la oportunidad de ganar en la última fecha contra el Inter.
pretender	El pretende salir vivo de esa cueva.

probablemente	Probablemente granice mañana.
probable	Es probable que alguien te llame el día de tu cumpleaños.
asumir	Asumí que la cena ya iba a estar servida.
incierto	Es incierta la hora en que va a morir.
incertidumbre	Hay incertidumbre por el resultado de las negociaciones.
desear	Siempre deseó comprar una casa a sus padres.
intención	Llegó a su casa con la intención de dormir.
aparentemente	Aparentemente el baño esta vació.
aparente	Causa alarma aparente lluvia de meteoros en California.
presumiblemente	El viento disipó las nubes que, presumiblemente, comenzaban a aparecer.
quizás	El hombre quizás descubra nuevos planetas.
hipotético	En el hipotético caso de que viniera, ya veríamos cómo acomodarla.
poder	Jaime podría encontrar la cura para la malaria.
inseguro	Fabio está inseguro sobre correr el triatlón de diciembre.
suponer	Fue caminando al trabajo por el supuesto paro de transporte.
supuestamente	Supuestamente nadie pudo abrir las persianas.

Palabras con implicancia + +

acordarse
adivinar
advertir
aprender
asimilar
averiguar
cerciorar
colegir
comprender
comprobar
concluir
confirmar
constatar
cuestionar
demostrar
desconocer
documentar
enterarse
estipular
evidenciar
ignorar
olvidar
probar
recordar
resolver
saber
sopesar
verificar
visualizar
convencer
evocar
persuadir
presentir
prever
perdonar
justificar
condenar
aprobar
criticar
reconocer
dejar
permitir
prohibir
tolerar
acostumbrar
aficionar

ayudar
favorecer
habituar
impeler
impulsar
incitar
inclinar
inducir
promover
forzar
imponer
obligar
poner
atrever
disponer
empeñar
apresurar
proceder
acertar
arriesgar
lograr

Palabras con implicancia + -

olvidar
negar
fingir
simular
hacer
creer
descartar
disuadir
fraguar
elucubrar
frustrar
impedir
imposibilitar
obstruir
pretender
evitar
negar
rehuir

Palabras con implicancia - +

olvidar
dudar
imaginar
soñar
creer
disuadir
dudar
esperar
estimar
imaginar
intuir
presumir
sospechar
suponer
titubear
vacilar
frustrar
impedir
imposibilitar
obstruir
pretender
evitar
rehuir

Palabras con implicancia - -

lograr
intentar
fingir
simular
hacer
creer
convencer
dejar
permitir
poner
intentar
tratar
atrever
disponer
procurar
proceder
acertar
arriesgarse