Detección de Especulaciones utilizando Active Learning

Informe de Proyecto de grado

Benjamín Machín Serna

Tutores:
MSc Diego Garat
Dr Guillermo Moncecchi

Diciembre, 2013

Resumen

Una especulación es una palabra o giro del lenguaje que permite suavizar una afirmación, dejando la posibilidad de que no sea cierta. Existe un tipo de especulaciones denominadas *weasel words*, que en general son utilizadas para crear la impresión de que algo específico y con significado se ha dicho, cuando en realidad se ha comunicado algo vago o ambiguo.

El objetivo de este trabajo consiste en estudiar el problema de la detección de oraciones que contienen *weasel words*, utilizando una metodología de aprendizaje automático semisupervisado denominada *active learning*. Esta metodología tiene la particularidad de dejar que el algoritmo de aprendizaje elija los ejemplos para entrenarse. En particular, se utilizan dos clasificadores basados en *Naive Bayes* y SVM, y se compara su performance al aplicar entrenamiento tradicional contra la performance que se obtiene utilizando *active learning*.

Los clasificadores construidos alcanzan resultados comparables aunque aún por debajo a los del estado del arte y la mejora que se logra al aplicar *active learning* no es significativa. Sin embargo, es necesario continuar con la investigación antes de descartar la utilización de *active learning* para la detección de oraciones con este tipo de especulaciones.

***Palabras clave*:** weasel words, active learning, naive bayes, svm.

Dedico este trabajo a mi familia y amigos por estar cerca y mostrarme en estos años cuáles son las cosas más importantes que hay que aprender. Agradezco también a mis tutores, Diego y Guillermo, que me han guiado y ayudado siempre con excelente disposición.

Índice general

Capítulo 1

Introducción

En los últimos años la comunidad del procesamiento del lenguaje natural (PLN) ha demostrado marcado interés por el estudio de las especulaciones. En su concepción más general, este fenómeno permite al interlocutor agregar a su discurso un componente de opinión, certeza, creencia y otros factores que de alguna forma debilitan el contenido factual de las proposiciones. El estudio de este aspecto del lenguaje es importante en cuanto a que la correcta detección de contenido especulativo permite mejorar los procesos de extracción de información haciendo posible la distinción de información factual de la que no lo es, para luego presentar los resultados en forma acorde.

Por otro lado, la aplicación de métodos de aprendizaje automático para la resolución de tareas de PLN ha crecido en las últimas décadas, gracias a una serie de factores como la disponibilidad de recursos lingüísticos anotados y el incremento constante en la capacidad de cómputo del hardware disponible para la investigación (Jurafsky and Martin, 2000).

Un factor limitante en este sentido es el alto costo de producción de material anotado. Surge entonces naturalmente la necesidad de desarrollar metodologías que eliminen o mitiguen este problema. Un acercamiento en esta línea es el *active learning*, un enfoque cuya principal idea es permitir que el algoritmo de aprendizaje elija activamente la información de la que aprende. En este proyecto se investiga la aplicabilidad del *active learning* al problema de la detección de especulaciones, comparando esta metodología con enfoques tradicionales.

En lo que sigue de este capítulo se describe con mayor detalle el tipo de fenómeno que se intenta detectar, se presenta la metodología a investigar para su resolución y se establecen los objetivos del trabajo.

Dado que este estudio se realizó sobre corpus en inglés, los ejemplos que aparecen a lo largo del documento se presentan en ese idioma. Además, se tomó la decisión de no traducir algunas expresiones para las cuales no nos

fue posible encontrar un equivalente en nuestro idioma que resulte cómodo sin alterar el significado original.

1.1. Weasel Words

La expresión "weasel words" se utiliza en inglés coloquial para denotar aquellas palabras o frases a través de las cuales se intenta crear la impresión de que algo específico y con significado se ha dicho, cuando en realidad solo se ha comunicado algo vago o ambiguo (Farkas et al., 2010). Este tipo de expresiones no dan cuenta de los hechos en forma neutral, sino que más bien expresan una opinión sin fundamento explícito o sin las referencias adecuadas. La siguiente oración es un ejemplo en donde la fuente de la información mencionada no se especifica, a la vez que se genera la sensación de que lo dicho es verdad en un sentido más bien general:

Ejemplo 1.1 (Weasel words)

- *Many argue* *that the news media are simply catering to public demand. ("Muchos argumentan que los medios de comunicación solo están alimentando la demanda pública.")* [1]

Las *weasel words* del ejemplo se destacan en el marco.

Los orígenes de la expresión se remontan al menos a principios del siglo pasado, siendo un artículo publicado en el New York Times en setiembre de 1916 uno de los primeros documentos escritos sobre el tema. El artículo relata el nacimiento de la expresión en un diálogo entre un nativo americano de nombre Dave Sewall y el entonces reciente ex-presidente de los Estados Unidos, Theodore Roosevelt, en el cual el primero, haciendo referencia a un hombre que acababan de cruzar en el camino, dice:

> "Ese hombre es capaz de hacer muchas cosas raras con nuestro lenguaje. Puede tomar una palabra y jugar con ella y succionar su contenido de la misma forma que una comadreja succiona el contenido de un huevo, hasta que ya no significa nada, sin importar cómo suene". [2]

[1] Una versión más clara de esta frase debería especificar quiénes son exactamente aquellos que mantienen ese argumento con respecto a los medios de comunicación.

[2] Transcripción original: "That there man can do a lot of funny things with this language of ours. He can take a word and weasel it around and suck the meat out of it like a weasel sucks the meat out of an egg, until it don't mean anything at all, no matter what it sounds like it means."

Más allá de su tono informal, esta frase captura algunas de las principales características que se utilizan hoy día para definir las *weasel words*, haciendo referencia explícita a la manipulación retórica del lenguaje con el fin de aparentar que lo expresado tiene un significado que no es realmente verdadero.

En Ganter and Strube (2009) se explica que la noción de *weasel words* está directamente relacionada con la de *hedging*, definida en Hyland (1998) como "cualquier medio lingüístico por el cual se expresa o bien una cierta falta de total compromiso con el valor de verdad de una proposición, o bien el deseo de no expresar ese compromiso categóricamente". Además comentan que una proporción importante de las expresiones de este tipo en Wikipedia se puede clasificar en tres categorías:

1. Sujetos numéricamente subespecificados: *"Most people agree that the drug war has failed."* ("La mayoría de las personas están de acuerdo en que la guerra contra las drogas ha fracasado.")

2. Construcciones que utilizan voz pasiva: *"It is known that the drug war has failed."* ("Es sabido que la guerra contra las drogas ha fracasado.")

3. Adverbios: *"Probably, the drug war has failed."* ("Probablemente, la guerra contra las drogas ha fracasado.")

A su vez, existen otras construcciones más complejas, aunque no menos frecuentes, que se consideran *weasel words*:

1. Falacias *non sequitur*, en las cuales una cierta conclusión no se deduce de las premisas. Por ejemplo:

   - Si soy un humano soy mamífero.
   - Soy un mamífero.
   - Por lo tanto, soy un humano.

2. Utilización de eufemismos. "La mesa directiva acordó una racionalización de las fuerzas de trabajo".

3. Generalizaciones imprecisas ("Algunas personas","Los expertos","Muchos").

Observamos que existe una heterogeneidad considerable en las expresiones y construcciones que se utilizan como *weasel words*, lo cual aumenta la dificultad de su correcta detección.

En particular, es de especial importancia para Wikipedia mantener sus artículos libres de *weasel words*, por lo que proveen un conjunto de recomendaciones orientadas a evitarlas, dado que "ayudan a oscurecer el significado de

expresiones no objetivas y son por lo tanto deshonestas"[3]. Además, facilitan a sus editores un conjunto de etiquetas y categorías para marcar artículos con *weasel words* para su posterior corrección. Las etiquetas aplican en distintos niveles de alcance, siendo posible utilizarlas para marcar algunas oraciones en particular dentro de un artículo, o, a nivel más general, anotando párrafos, secciones o artículos completos para corregir.

Durante el desarrollo de este trabajo nos enfocaremos en las definiciones y recomendaciones de Wikipedia, debido a que el corpus utilizado en los experimentos se compone de una selección de párrafos extraídos de ella.

1.2. ¿Por qué Active Learning?

Distinguimos tres formas de clasificar los métodos de aprendizaje automático. Por un lado, están aquellos métodos que utilizan datos etiquetados para aprender una función de clasificación, categorizados como métodos de aprendizaje supervisado. Por otro, están los métodos de aprendizaje no supervisado, que sólo utilizan datos sin etiquetar en su aprendizaje. En medio se encuentran los métodos de aprendizaje semisupervisado, correspondientes a los enfoques híbridos entre los dos anteriores, donde además, típicamente, la proporción de datos no anotados es mayor que la de los datos anotados.

El *active learning* es un método de aprendizaje semisupervisado, que propone minimizar el costo asociado al proceso de inferencia de las funciones de clasificación permitiendo al algoritmo de aprendizaje ser "curioso". La idea es que, si el algoritmo tiene la posibilidad de elegir libremente los ejemplos etiquetados de los que aprende, los resultados serán mejores, a la vez que se reduce el costo de aprendizaje (Settles, 2009).

Uno de los principales atractivos del *active learning* es la forma en que se adapta a la vasta disponibilidad de información no procesada existente en la actualidad (en internet por ejemplo) en contraposición a su alto costo de procesamiento. Esto se logra mediante un método de construcción de clasificadores que permite elegir interactivamente los ejemplos de aprendizaje, con el objetivo de minimizar la intervención de expertos para obtener los ejemplos etiquetados.

La aplicación exitosa de *active learning* en el dominio de estudio de este trabajo implicaría obtener clasificadores para detectar *weasel words* con performance competitiva y bajos costos de anotación humana. Por ejemplo, se podría confeccionar un corpus de entrada parcialmente anotado directamente a partir de respaldos (llamados *dumps*) de Wikipedia, para que luego

[3]Recomendaciones sobre *weasel words* de Wikipedia: `http://simple.wikipedia.org/wiki/Wikipedia:Avoid_weasel_words`. Último acceso el 17 de Julio de 2013

el algoritmo seleccione los mejores ejemplos para aprender. Esta forma de trabajo permitiría reducir la cantidad ejemplos a ser anotados por expertos humanos.

1.3. Objetivos

El objetivo principal de este proyecto es la evaluación de una metodología de aprendizaje semisupervisado llamada *active learning* y su aplicación al problema de la detección de oraciones inciertas, en particular a la detección de oraciones que contienen un tipo de expresiones denominadas *weasel words*. Determinar con exactitud estas expresiones dentro de una oración no es siempre una tarea simple, por lo cual se opta por reducir el problema a la identificación de su existencia en alguna posición de la oración. Esta tarea se llevará a cabo mediante la construcción de clasificadores que utilicen *active learning*, y, con el fin de evaluar los resultados, se comparará su performance con la de clasificadores tradicionales de aprendizaje supervisado.

Además, resulta necesario contar con una solución competitiva con respecto al estado del arte que resuelva el problema de la detección de oraciones con *weasel words*, de forma de tener un punto de partida razonable para la comparación de los enfoques.

1.4. Organización del documento

El resto del documento se organiza como sigue:

En el capítulo 2 se presentan los conceptos de aprendizaje automático aplicados durante el desarrollo del proyecto, incluyendo una descripción de los algoritmos tradicionales empleados y una breve introducción al *active learning*.

En el capítulo 3 se resume el estado del arte del problema de la detección de especulaciones, y en particular el estado del arte de la detección de oraciones con *weasel words*. Luego se describe la aproximación computacional adoptada para resolver el problema, se introducen los corpus utilizados en el trabajo, se describen las diferentes pruebas realizadas, y se detalla el proceso de desarrollo de cada una. Finalmente se presentan algunos detalles de implementación.

Los resultados de las pruebas se presentan en el capítulo 4, junto con el análisis de los errores cometidos por los clasificadores desarrollados. En este capítulo también se realiza la comparación entre los resultados que se obtuvieron con los enfoques tradicionales y con *active learning*.

Por último, en el capítulo 5 se presentan las conclusiones del trabajo, y los posibles puntos de mejora y extensión.

Capítulo 2

Aprendizaje automático

En este capítulo se presentan las metodologías de aprendizaje automático utilizadas durante los experimentos llevados a cabo para cumplir con los objetivos del proyecto. En primer lugar se describen las dos aproximaciones tradicionales empleadas (*Naive Bayes* y Máquinas de Vectores de Soporte), y luego se presenta una introducción al *active learning*, abarcando sus principales escenarios y estrategias. Se cierra el capítulo con un breve estudio de casos de aplicación práctica de *active learning*.

2.1. Aprendizaje automático y clasificación

Un problema de clasificación consiste en asignar categorías o clases a elementos de un dominio dado. Para la construcción del clasificador, se puede partir de un conjunto de ejemplos ya etiquetados y darlos a un algoritmo que se *entrena* con ellos y *aprende* a clasificar; se dice que un programa aprende si logra mejorar su performance respecto a una medida de evaluación dada a través de la experiencia (Mitchell, 1997). En definitiva, luego del entrenamiento, el clasificador obtenido es capaz de etiquetar nuevos ejemplos.

Se puede entender a estos métodos de aprendizaje como métodos de generalización, donde a partir de un conjunto inicial se deducen las características de los ejemplos de cada clase. Las características que se tienen en cuenta para cada ejemplo se denominan *atributos*, y es crucial elegirlos de forma que permitan el aprendizaje, teniendo tanto la capacidad de distinguir ejemplos entre sí como la de agruparlos en clases.

A modo de ejemplo, si se tiene información de personas, los atributos adecuados para determinar el público objetivo de una campaña publicitaria de perfumes y los que se consideran para identificar la población más propensa a una enfermedad cardiovascular son bien diferentes. Si bien hay

atributos que pueden ser útiles en ambos casos (como la edad y el género de los individuos), existen atributos específicos para cada dominio de interés que pueden resultar beneficiosos en un caso pero ineficaces en el otro (historial de consumo, antecedentes familiares).

Presentamos a continuación dos modelos de clasificadores ampliamente utilizados en el área, que fueron los elegidos para los experimentos realizados en este trabajo.

2.2. Naive Bayes

El modelo de clasificadores de Bayes "ingenuo" o *Naive Bayes* es un modelo clásico en aprendizaje automático, conocido por su simplicidad y performance competitiva con respecto a modelos actuales. El modelo se dice ingenuo debido a que asume que los atributos de cada ejemplo son independientes entre sí, lo cual constituye una premisa fuerte y poco frecuente en la realidad. Sin embargo, el modelo alcanza buenos resultados, y ha sido aplicado con éxito en áreas como clasificación de documentos y diagnosis médica (Mitchell, 1997).

La base del modelo está en el teorema de Bayes, que nos permite calcular la probabilidad condicional de un evento A sujeto a la ocurrencia de otro evento B, según:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

En nuestro caso, queremos saber la probabilidad de que una oración sea incierta o no, a partir de la información que nos brindan los atributos elegidos (que se presentan más adelante). En otras palabras, queremos saber cuál es la etiqueta más probable para una oración representada por los valores de sus atributos:

$$P(etq|a_1 = v_1, a_2 = v_2, ..., a_n = v_n) = \frac{P(a_1 = v_1, a_2 = v_2, ..., a_n = v_n|etq)P(etq)}{P(a_1 = v_1, a_2 = v_2, ..., a_n = v_n)}$$

donde:

$$etq \in \{certain, uncertain\}$$

y $a_i = v_i$ es el evento que tiene lugar cuando el atributo i-ésimo toma el valor v_i. Queremos conocer la etiqueta etq^* que maximiza dicha probabilidad, por lo cual podemos despreciar el denominador ya que es independiente de la etiqueta que se considere. Entonces:

$$etq^* = \arg\max_{etq} P(a_1 = v_1, a_2 = v_2, ..., a_n = v_n|etq)P(etq)$$

Por otro lado, utilizando la definición de probabilidad condicional y despejando:

$$P(etq, a_1 = v_1, a_2 = v_2, ..., a_n = v_n) = P(a_1 = v_1, a_2 = v_2, ..., a_n = v_n|etq)P(etq)$$

Por lo cual:

$$etq^* = \arg\max_{etq} P(etq, a_1 = v_1, a_2 = v_2, ..., a_n = v_n)$$

El modelo ingenuo simplifica este cálculo asumiendo que los eventos asociados a los valores de los atributos son independientes, de forma que:

$$P(a_1 = v_1, a_2 = v_2, ..., a_n = v_n|etq) = \prod_{i=1}^{i=n} P(a_i = v_i|etq)$$

Utilizando este resultado y la definición de probabilidad conjunta, podemos reescribir:

$$etq^* = \arg\max_{etq} P(etq) \prod_{i=1}^{i=n} P(a_i = v_i|etq)$$

Finalmente, para aproximar $P(etq)$ y $P(a_i = v_i|etq)$ utilizamos los estimadores de máxima verosimilitud:

$$P(etq) = \text{proporción de oraciones etiquetadas con } etq$$
$$P(a_i = v_i|etq) = \frac{\text{\# oraciones etiquetadas con } etq \text{ en las que } a_i = v_i}{\text{\# de oraciones etiquetadas con } etq}$$

De esta forma, partiendo de un conjunto de entrenamiento conformado por oraciones etiquetadas, se tiene una forma de aproximar la probabilidad de cada etiqueta para cualquier oración, dados los valores de sus atributos.

2.3. Máquinas de Vectores de Soporte

Resulta importante contar con resultados de otro tipo de clasificadores con el objetivo de poder comparar sus performances. Las Máquinas de Vectores de Soporte (SVM del inglés *Support Vector Machines*) constituyen un modelo de aprendizaje que ha demostrado una gran capacidad de aplicación práctica en problemas de clasificación. En particular, algunos de los mejores resultados para las tareas de la *Shared Task* de la ConLL 2010 (ver capítulo 3) se atribuyen a clasificadores que utilizan SVM (Georgescul (2010); Morante et al. (2010); Tang et al. (2010)).

Además, este modelo ofrece la posibilidad de comparar el enfoque generativo de *Naive Bayes* contra un modelo discriminativo, lo que lo convierte en un buen candidato para implementar el segundo clasificador del experimento.

Presentamos a continuación una breve introducción a los fundamentos matemáticos de este modelo basada en (Boswell, 2002).

Dado un conjunto de n ejemplos de entrenamiento de la forma $\{x_i, y_i\}$ con $i = 1..n$, donde cada instancia x_i es un vector de p dimensiones ($x_i \in \mathbb{R}^p$) y tiene asociada una etiqueta ($y_i \in \{1, -1\}$), buscamos un hiperplano que divida el espacio $\mathbb{R}^p$ en dos regiones, separando las instancias x_i según sus etiquetas. Asumimos que tal plano existe, es decir, que los datos son linealmente separables. De esta forma, clasificaremos nuevos puntos según la región del espacio a la que pertenezcan.

Consideramos la ecuación paramétrica de los hiperplanos de $\mathbb{R}^p$:

$$w \cdot x + b = 0$$

Contando con $(\hat{w}, \hat{b})$ tales que la ecuación representa un hiperplano que separa correctamente las instancias x_i, obtenemos la función de clasificación:

$$f(x) = signo(\hat{w} \cdot x + \hat{b})$$

Dado que el plano determinado por $(\hat{w}, \hat{b})$ es el mismo que el determinado por $(\lambda\hat{w}, \lambda\hat{b})$ con $\lambda \in \mathbb{R}$, se considera la *distancia funcional* de un punto (x_i, y_i) al plano dado por (w, b) como:

$$y_i(x_i \cdot w + b)$$

y se definen los *hiperplanos canónicos* como aquellos hiperplanos que cumplen:

$$y_i(x_i \cdot w + b) \geq 1, \forall i = 1..n$$

Se llaman *vectores de soporte* a los puntos que cumplen:

$$y_i(x_i \cdot w + b) = 1$$

Vale la pena notar que la distancia funcional es distinta a la distancia Euclidiana habitual. La distancia geométrica habitual de un punto x_i al plano dado por (w, b) es:

$$d((w, b), x_i) = \frac{y_i(x_i \cdot w + b)}{\|w\|}$$

Observamos que la distancia geométrica de un punto al plano dado por $(\lambda\hat{w}, \lambda\hat{b})$ se mantiene independientemente del λ, pero la distancia funcional cambia según este parámetro. La idea intuitiva de las SVM es buscar aquel hiperplano que maximice la distancia geométrica a los vectores de soporte (que son los puntos más cercanos según la distancia funcional). Esto se logra

minimizando $\|w\|$, sujetos a las restricciones impuestas en la definición de los hiperplanos canónicos:

$$\begin{aligned} &\underset{w}{\text{minimizar}} && \|w\| \\ &\text{sujeto a} && y_i(x_i \cdot w + b) \geq 1, \forall i \in 1..n \end{aligned}$$

De estas ecuaciones se desprende que los únicos puntos relevantes para determinar el hiperplano óptimo son los vectores de soporte (Vapnik, 1995).

Existen extensiones al modelo para aplicar en casos en que los ejemplos de entrenamiento no son linealmente separables, que es el caso más frecuente en la realidad. La idea de estas extensiones es aplicar transformaciones en las que el problema es llevado a dimensiones más altas, donde sí es posible separar los ejemplos linealmente. Para obtener información más detallada referirse a Vapnik (1995).

2.4. Active Learning

El *active learning* es un método de aprendizaje automático que busca reducir los costos de construcción de clasificadores a través de la selección interactiva e inteligente de los ejemplos de entrenamiento. De esta forma se intenta minimizar la cantidad de trabajo de anotación, manteniendo o incluso mejorando la performance de los clasificadores obtenidos.

El diagrama de la figura 2.1 ilustra el ciclo de entrenamiento que utiliza el *active learning*. Se parte de un conjunto inicial de ejemplos etiquetados, a partir del cual se entrena un clasificador. Luego se debe elegir o generar un nuevo ejemplo (no etiquetado) que será agregado al conjunto etiquetado, esperando que esto maximice la mejora en la performance del clasificador. Una vez elegido, el ejemplo se entrega al oráculo, que es en general un anotador experto, que nos da la etiqueta correspondiente. Finalmente, se agrega el nuevo ejemplo etiquetado al conjunto inicial, y se repite el entrenamiento. El proceso termina cuando se alcanza un umbral de performance preestablecido o cuando se agotan los ejemplos no etiquetados (en el peor caso). En general, el método aplica en casos en que el costo de consultar al oráculo es alto, y se parte de un conjunto inicial relativamente pequeño.

Las formas en que el algoritmo recibe la información y las estrategias con las que selecciona los ejemplos de los cuales aprende son diversas. Por ejemplo, se puede partir de una muestra grande de instancias sin etiquetar de la cual se seleccionarán las "interesantes". En otro escenario, los ejemplos se pueden recibir secuencialmente decidiendo en el momento si es necesario consultar su etiqueta, y en algunos casos es posible directamente sintetizar los

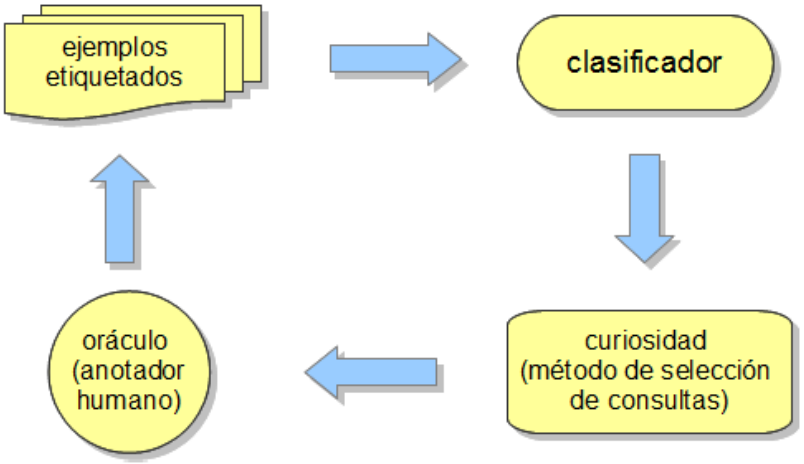

Figura 2.1: Ciclo de entrenamiento del active learning.

ejemplos interesantes para que el oráculo clasifique. Por otro lado, la selección de los ejemplos a consultar se puede basar, por ejemplo, en la probabilidad de error del modelo, el nivel de desacuerdo entre los modelos consistentes con el conjunto de entrenamiento, o combinaciones. En las siguientes secciones entramos en detalle de los diferentes escenarios y estrategias más comúnmente utilizados. Se toman como base los trabajos presentados en Settles and Craven (2008) y Settles (2009).

2.4.1. Síntesis de consultas

Uno de los escenarios más comunes es aquel en que las instancias a etiquetar por el oráculo son sintetizadas por el algoritmo de aprendizaje. Se intenta crear ejemplos que maximicen el aporte de información al modelo. Si el dominio del que provienen los datos a clasificar es conocido, existen en general estrategias de síntesis que funcionan razonablemente bien. Por ejemplo, si se está aprendiendo un clasificador binario, se intentará sintetizar ejemplos para los cuales ambas etiquetas sean equiprobables, dado que esto permite reducir a la mitad la cantidad de clasificadores consistentes con los ejemplos conocidos.

Un posible problema con este enfoque en dominios no finitos es que, al sintetizar consultas automáticamente, se pueden generar ejemplos extraños que podrían inducir errores del oráculo. Por ejemplo, si se están clasificando imágenes, el etiquetado de una imagen generada por el algoritmo puede resultar desconcertante para un anotador humano, posiblemente provocando errores.

2.4.2. Flujo de muestras

Cuando las instancias llegan como un flujo de datos sin etiquetar, o se puede simular este comportamiento, se puede evitar el problema de la generación de consultas arbitrarias potencialmente desconcertantes, ya que los ejemplos son reales. Cada instancia que llega secuencialmente es analizada y el algoritmo decide si consulta al oráculo o simplemente la ignora y pasa a la siguiente. Bajo este escenario asumimos que la obtención de datos no etiquetados es barata, de forma tal que el hecho de descartar ejemplos no implique pérdidas computacionales importantes. Cuando la distribución de entrada es uniforme, este escenario es similar al anterior, ya que cualquier instancia tiene la misma probabilidad de ocurrir y ser elegida por el algoritmo (incluso aquellas desconcertantes para el oráculo). Sin embargo, cuando la distribución no es uniforme, este enfoque asegura que las muestras obtenidas serán razonables y representativas, en cuanto a que responden a la distribución real de las instancias.

2.4.3. Conjuntos de muestras

Este escenario se da cuando se tiene acceso a un extenso conjunto sin anotar cuyos ejemplos pueden ser accedidos en conjunto, en contraposición al escenario anterior, donde se obtiene de a una instancia por vez. Las consultas a realizar son seleccionadas de este conjunto teniendo en cuenta todos sus ejemplos, lo que permite una mejor performance. Si bien este escenario resulta atractivo, en algunos casos el costo de evaluar todos los ejemplos en cada paso puede llevar a que sea más conveniente simular el escenario de flujo de muestras o, al menos, acelerar el proceso tomando pequeños subconjuntos de ejemplos por vez.

2.4.4. Estrategias de selección de consultas

Para poder seleccionar inteligentemente las instancias a consultar, el algoritmo de aprendizaje debe tener una forma de calificar o medir la importancia de conocer la etiqueta correcta de una instancia. Luego, según esta estimación se seleccionará aquella instancia cuya etiqueta aporte mayor información al modelo en construcción. Una de las estrategias principales considera que la instancia más valiosa para el aprendizaje es aquella para la cual la probabilidad estimada de obtener una predicción correcta es la mínima. La segunda estrategia que presentamos seleccionará aquella instancia que maximice el nivel de desacuerdo de un conjunto preseleccionado de clasificadores consistentes con el conjunto de etiquetas conocidas. Veremos además mejoras sobre

ambas estrategias, y comentaremos brevemente otros enfoques.

Consultas por incertidumbre

La selección por incertidumbre es uno de los modelos más comunes para medir la información de las etiquetas, y consiste en seleccionar aquella instancia para la cual el algoritmo tiene menos seguridad de predicción. Es decir, el algoritmo tratará de seleccionar aquella instancia que maximice la probabilidad de que su etiqueta asociada sea distinta a la predicha. En casos de clasificación binaria, esta estrategia es simple y consiste en elegir aquella instancia cuya probabilidad de tomar uno de ambos valores sea más cercana a 0,5. En casos más generales (con mayor cantidad de etiquetas) se define la instancia menos confiable como:

$$x^*_{LC} = \underset{x}{\text{argmax}}\ 1 - P_\theta(\hat{y}|x)$$

donde

$$\hat{y} = \underset{y}{\text{argmax}}\ P_\theta(y|x)$$

es la etiqueta más probable para x según el modelo θ.

Existen mejoras para el caso general que no discutiremos aquí, ya que no son aplicables cuando se trabaja con clasificación binaria. Para profundizar en el tema se sugiere recurrir a Settles (2009).

Consultas por desacuerdo

La intuición detrás de este modelo (Seung et al., 1992) consiste en elegir aquella instancia para la cual existe mayor desacuerdo entre las predicciones de los clasificadores consistentes con los ejemplos conocidos. Dado el conjunto actual de instancias etiquetadas conocidas, y un subconjunto de clasificadores consistentes con él, que llamamos "comité", se elige de entre las instancias no etiquetadas aquella instancia que genera mayor desacuerdo entre ellos. Cada nueva etiqueta obtenida reduce el conjunto de clasificadores candidatos, hasta obtener un conjunto suficientemente pequeño para satisfacer cierto criterio de parada.

Para poder aplicar esta estrategia debemos ser capaces de construir un comité de clasificadores a partir de las hipótesis conocidas, y encontrar una forma de medir el desacuerdo entre ellos. Una forma simple de construir este conjunto es elegirlos aleatoriamente del conjunto de clasificadores candidatos (todos aquellos clasificadores en el espacio solución que son consistentes con las hipótesis conocidas). Por otro lado, no existe acuerdo sobre el tamaño óptimo del comité, que en general depende del problema a resolver.

Presentamos ahora dos de los enfoques para medir el nivel de desacuerdo entre los miembros que son más frecuentes en la literatura. El primero (Dagan and Engelson, 1995) considera la probabilidad de cada etiqueta como la suma de los "votos"que ésta obtiene en relación al tamaño del comité, eligiendo la instancia x que tenga máxima entropía, es decir:

$$P(y_i|x) = \frac{V(y_i)}{C}$$

donde $V(y_i)$ es la cantidad de votos recibidos por la etiqueta y_i para x, y C es el tamaño del comité. De esta forma obtenemos:

$$x^*_{VE} = \underset{x}{\text{argmax}} \; -\sum_i \frac{V(y_i)}{C} \log \frac{V(y_i)}{C}$$

En segundo lugar, consideraremos la *divergencia Kullback-Leibler* promedio (McCallum and Nigam, 1998), que utiliza la noción de diferencia entre dos distribuciones de probabilidad, eligiendo aquella instancia para cual la diferencia entre la distribución de etiquetas entre un miembro del comité y el consenso (todo el comité) es máxima. Si llamamos $P_{\theta^{(C)}}(y_i|x)$ a la probabilidad de la etiqueta y_i para el ejemplo x según el miembro c del comité y análogamente llamamos $P_C(y_i|x)$ a la probabilidad consensuada de la etiqueta y_i para el ejemplo x, se define la diferencia entre ambas distribuciones como:

$$D(P_{\theta^{(C)}}||P_C) = \sum_i P_{\theta^{(C)}}(y_i|x) \log \frac{P_{\theta^{(C)}}(y_i|x)}{P_C(y_i|x)}$$

obteniendo así el siguiente algoritmo de selección:

$$x^*_{KL} = \underset{x}{\text{argmax}} \frac{1}{C} \sum_{c=1}^{C} D(P_{\theta^{(C)}}||P_C)$$

Observando que la probabilidad consensuada de la etiqueta y_i para x es:

$$P_C(y_i|x) = \frac{1}{C} \sum_{c=1}^{C} P_{\theta^{(C)}}(y_i|x)$$

Otras estrategias

Existen estrategias de selección más complejas que contemplan escenarios donde el oráculo es imperfecto o donde el espacio de hipótesis no contiene un miembro óptimo (el problema no es perfectamente realizable por el espacio de hipótesis considerado). Dado que estas dificultades escapan al alcance del proyecto, solo comentaremos brevemente dos estudios en esta dirección.

Ponderación por importancia Esta estrategia (Beygelzimer et al., 2009) toma los ejemplos de a uno por vez (flujo de muestras), y basado en la historia de los ejemplos anteriores no etiquetados, las etiquetas conocidas, y el ejemplo x_t en cuestión, se calcula una probabilidad p_t de consultar al oráculo por la etiqueta y_t. Se mantiene además el conjunto de etiquetas conocidas junto con una medida de su importancia, que corresponde al valor $1/p_t$, lo cual permite evitar el sesgo de los ejemplos consultados. En Beygelzimer et al. (2009) se demuestra que este método garantiza la convergencia del algoritmo al miembro óptimo del espacio de hipótesis, y que la cantidad de etiquetas consultadas en el peor caso no es mucho mayor que la cantidad consultada en aprendizaje supervisado.

Active learning agnóstico El active learning agnóstico (Beygelzimer et al., 2010) es una variante de la estrategia anterior, donde se propone no mantener un espacio solución en cada paso. Esto permite mejorar la robustez del algoritmo, ya que se evitan problemas que pueden surgir si por errores de modelado o aproximaciones se descarta del espacio solución el clasificador que obtiene el resultado óptimo. La principal diferencia está en que en cada paso t, se utilizan las dos mejores hipótesis dentro de todo el espacio solución que minimizan el error con respecto a las etiquetas conocidas para asignar la probabilidad p_t. Se demuestra (Beygelzimer et al., 2010) que con esta estrategia, además de tener consistencia (siempre se obtiene la hipótesis óptima), se sigue manteniendo una mejora sobre métodos de aprendizaje tradicionales.

2.4.5. Aplicación práctica de Active Learning

Existe una cantidad considerable de publicaciones teóricas sobre *active learning* donde se examinan las ventajas de la metodología frente a otros enfoques, en escenarios ideales. Sin embargo, no es tan abundante el número de publicaciones que presentan aplicaciones prácticas reales de este enfoque. A continuación se referencian algunos trabajos que presentan resultados positivos en el área.

Lewis and Gale (1994) es una de las primeras publicaciones en donde se muestra la eficacia de la selección de consultas por incertidumbre, una técnica típica de *active learning*. En su experimento se clasifican noticias a partir de sus títulos, y se logra reducir la cantidad de ejemplos etiquetados de 100000 a 1000 manteniendo un desempeño competitivo.

En Zhu et al. (2003) se aplica *active learning* utilizando campos aleatorios Gaussianos para elegir los ejemplos a etiquetar según se minimice el error esperado de clasificación. Su enfoque resulta exitoso en el reconocimiento

de escritura de dígitos y en la clasificación binaria de mensajes de correo electrónico.

Tong and Chang (2001) utilizan exitosamente *active learning* con máquinas de vectores de soporte para la clasificación binaria de imágenes. Tur et al. (2005) combinan *active learning* con aprendizaje semisupervisado para clasificar lenguaje hablado, reduciendo a la mitad la cantidad de ejemplos etiquetados sin afectar la performance.

Además del trabajo en estas publicaciones, más recientemente, en Settles (2011) se menciona que compañías de software como Google, IBM, Microsoft y Siemens están utilizando *active learning* en la construcción de sus aplicaciones.

Por otro lado, a pesar de la mejoras de performance y reducción de costos de anotación que promete, el *active learning* no ha alcanzado aún el nivel de aplicación práctica que sí tienen otras metodologías de aprendizaje automático. Esto se debe a que muchas de las premisas que se asumen en los trabajos teóricos son rápidamente quebrantadas al llevar las ideas a la práctica (Settles, 2011). En nuestro caso, experimentamos con el uso de esta metodología aplicada en un problema de interés actual, que tampoco ha alcanzado los niveles de performance de otros problemas tradicionales de PLN.

Capítulo 3

Detección de Weasel Words con Active Learning

En este capítulo se presenta en primer lugar el estado del arte del problema de la detección de especulaciones, en particular, la detección de oraciones con *weasel words*. Seguidamente se presenta el abordaje computacional adoptado y los corpus sobre los que se trabajará. Luego se dedica una sección a la descripción del diseño del experimento, es decir, las diferentes pruebas a realizar de forma de cumplir con los objetivos del trabajo. y se dedica una sección a la presentación de los criterios de evaluación. Se prosigue con la descripción de los primeros modelos implementados (Naive Bayes y SVM) y su proceso de mejora. Finalmente, se presenta el trabajo realizado con *active learning*, describiendo el enfoque adoptado para el escenario del proyecto y el algoritmo aplicado. La última sección se dedica a presentar algunos detalles de implementación.

3.1. Detección de especulaciones

La calidad especulativa de una proposición viene dada por lo que se llaman "marcas de especulación", del inglés *hedge cues* (Lakoff, 1973), que son aquellas palabras que debilitan la seguridad o certeza de una proposición. Las marcas de especulación pueden ocurrir como palabras aisladas (*"seems"*, *"suggest"*, *"maybe"*), secuencias de palabras (*"indicates that"*, *"there is no evidence"*) o incluso como secuencias discontinuas de palabras. Por otro lado, si bien existen palabras que típicamente forman parte de marcas de especulación, éstas no son siempre empleadas con una semántica especulativa, como en el siguiente ejemplo:

Ejemplo 3.1 (Weasel words según contexto)

- *We suggest you bring your outerwear. ("Sugerimos llevar ropa de abrigo.")*
- *Climate theories crumble as data and experts suggest global cooling. ("Las teorías sobre el clima se derrumban frente a datos y expertos que sugieren enfriamiento global.")*

La correcta detección de estas marcas resulta entonces una tarea no trivial que va más allá de la simple categorización de palabras y requiere en general la consideración del contexto en que ocurren.

Una tarea más ambiciosa que extiende la detección de las marcas en sí, es la identificación de su *alcance* dentro de una oración. Es decir, la porción de información que es afectada por el carácter especulativo de cada marca. Si bien esta tarea es de interés para el área, escapa a los alcances de este proyecto. Para mayor información se sugiere recurrir a las referencias citadas, en particular a Morante and Sporleder (2012).

La mayoría de las publicaciones que tratan el tema de las especulaciones se centran en textos sobre biología molecular, y hacen uso del corpus BioScope (Szarvas, 2008). Es en Ganter and Strube (2009) que se introduce un dominio de análisis distinto en el que se busca detectar información no factual en oraciones de Wikipedia. En su trabajo experimentan con dos clasificadores, uno que se basa en las palabras que preceden a las *weasel words* y otro basado en patrones sintácticos. Ambos clasificadores obtienen resultados similares. El clasificador basado en patrones sintácticos obtiene mejores resultados en un conjunto de datos reanotados por los autores, sugiriendo que este enfoque puede detectar *weasel words* que no han sido detectadas anteriormente.

La tarea de la clasificación binaria de oraciones según su contenido especulativo se termina de consolidar en la *Shared Task* de la decimocuarta edición de la *Conference on Computational Natural Language Learning* (CoNLL) en 2010. Las *Shared Tasks* de estas conferencias consisten en la presentación de algún problema de actualidad en el área, el cuál se especifica en formato de competencia.

El título de la competencia en ese año fue *"Learning to Detect Hedges and their Scope in Natural Language Text"* (Farkas et al., 2010) y la primera tarea consistió en la detección de oraciones con contenido especulativo (clasificación binaria) en dos conjuntos de datos: artículos de biología molecular del BioScope corpus (Task1B), y párrafos de Wikipedia (Task1W). Los enfoques tomados para esta tarea pueden agruparse a grandes rasgos en dos categorías. Por un lado están aquellos sistemas que utilizaron técnicas de desambiguación de oraciones aplicando un enfoque basado en *Bag of Words*

(BOW), donde se ponderan las palabras según su cantidad de ocurrencias, sin importar el orden lineal en que éstas aparecen. Siguiendo otro enfoque están aquellos sistemas que se centraron en la detección de marcas de especulación, clasificando cada oración según contengan marcas o no. Dentro de esta segunda categoría, algunos sistemas utilizaron clasificación de las palabras basándose en *tokens*, mientras que otros utilizaron técnicas de clasificación secuencial, típicamente Campos Aleatorios Condicionales (Lafferty et al., 2001) (CRF, del inglés *Conditional Random Fields*).

En Farkas et al. (2010) se remarca el hecho de que los sistemas con mejores resultados en la tarea Task1W emplean el enfoque BOW, mientras que los mejores resultados en la tarea Task1B son logrados con enfoques de clasificación secuencial. Una posible causa es el hecho de que los textos sobre biología molecular siguen patrones simples, haciendo fácil el aprovechamiento del contexto en que las palabras ocurren tomando ventanas pequeñas, mientras que los párrafos de Wikipedia son de naturaleza mucho más diversa. El mejor sistema para la tarea Task1B corresponde a Tang et al. (2010), que utiliza un enfoque en cascada en el cual se utilizan dos CRFs. Sus resultados alcanzan 0,864 en medida F[1]. En el corpus de Wikipedia obtienen 0,550 en medida F, reafirmando lo mencionado anteriormente. Por otro lado, el mejor sistema para la tarea Task1W se presenta en Georgescul (2010) obteniendo 0,602 en medida F, alcanzando 0,785 puntos en medida F en la tarea Task1B.

3.2. Aproximación computacional

En nuestro caso, abordamos el problema de la detección de *weasel words* en una oración como un problema de *clasificación binaria*. Es decir, buscamos una función que, dada una oración, sea capaz de asignarle una de dos *etiquetas*, según incluya *weasel words* o no. A este tipo de funciones se las denomina *clasificadores*.

En general, previo a la clasificación de un ejemplo, es preciso realizar la *extracción de atributos*; llevar el ejemplo de entrada a una representación uniforme en donde se conocen los valores que toman los *atributos* elegidos. Los atributos son las características del ejemplo que serán considerados relevantes para decidir su categoría. En este caso, los ejemplos de entrada son oraciones y los atributos que consideramos están fuertemente ligados las palabras que ocurren en cada oración.

El esquema general de funcionamiento de un algoritmo de aprendizaje supervisado se muestra en la figura 3.1. Se parte de un conjunto de ejem-

[1]La medida F es una medida de resumen de las medidas de precisión y exhaustividad (*recall*). Estas definiciones se presentan más adelante en este capítulo.

plos etiquetados, y se extraen los valores de los atributos de cada ejemplo, obteniendo una representación uniforme. Esta información se entrega al algoritmo de aprendizaje, que construye un clasificador capaz de etiquetar nuevos ejemplos, que son previamente llevados a la misma representación uniforme.

Nuestro trabajo consiste entonces en elegir los atributos de las oraciones, implementar la función de extracción, y construir algoritmos capaces de generar o *entrenar* clasificadores a partir de los ejemplos etiquetados de entrada.

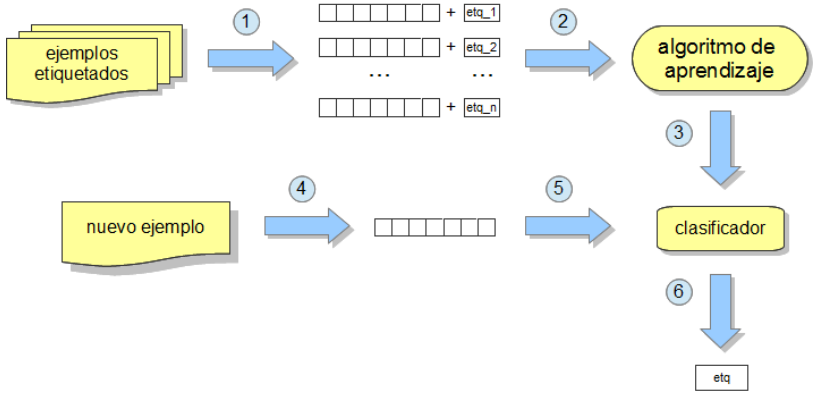

Figura 3.1: Aprendizaje supervisado. Extracción de atributos (1). El algoritmo de aprendizaje construye un clasificador (2 y 3). Llega un nuevo ejemplo, y se repite el proceso de extracción de atributos (4). El clasificador predice una etiqueta para el nuevo ejemplo (5 y 6).

3.3. Corpus de entrenamiento y evaluación

Con fines comparativos, los corpus utilizados en este trabajo son los mismos que se elaboraron para la primera tarea de la *Shared Task* de la ConLL 2010. A continuación se describe el proceso de extracción, según se explica en Farkas et al. (2010).

El corpus de entrenamiento fue confeccionado a partir de 2186 artículos de Wikipedia y cuenta con un total de 11111 oraciones de las cuales 2484 tienen contenido especulativo. A partir de otros 2346 artículos se construye el corpus de evaluación con 9634 oraciones conteniendo 2234 oraciones inciertas. Para la selección de los artículos se tuvo en cuenta el etiquetado por parte de los editores de Wikipedia que indica la presencia de *weasel words*. Cabe destacar que la cantidad de etiquetas utilizadas con este fin en Wikipedia no es menor, dado que existe un gran número de etiquetas equivalentes, a la vez

```
<sentence certainty="uncertain" id="S1.2">
  <ccue>Many argue</ccue> that the news media are simply catering
     to public demand.
</sentence>
```

Cuadro 3.1: El ejemplo 3.2 en el formato utilizado en la Shared Task de la ConLL 2010.

Corpus	Artículos	Oraciones factuales	Oraciones inciertas	Total
Entrenamiento	2186	8627	2484	11111
Evaluación	2346	7400	2234	9634

Cuadro 3.2: Resumen de los corpus utilizados.

que se pueden utilizar etiquetas a nivel de oración, de párrafo, de sección o incluso a nivel de artículo.

Ejemplo 3.2 (Formato de Wikipedia)
Many argue {*weasel-inline*} *that the news media are simply catering to public demand.*

A partir de un respaldo de la Wikipedia en inglés se extrajeron todos los artículos que incluían alguna etiqueta de *weasel*, alcanzando un total de 5874 etiquetas diferentes (Farkas et al., 2010). Luego se seleccionaron 438 de estos artículos en forma aleatoria para ser anotados manualmente. De este conjunto inicial se extrajo una lista de las *weasel words* más frecuentes, que se tomó como referencia para obtener el resto de los artículos. Finalmente todas las oraciones fueron anotadas manualmente, identificando como inciertas a aquellas oraciones que incluían al menos una etiqueta de *weasel*. El resultado de este procedimiento son dos corpus en los cuales aparecen palabras utilizadas como *weasel words* tanto en oraciones no especulativas como en oraciones inciertas. Ambos se encuentran publicados en formato XML en el sitio de la conferencia[2].

El ejemplo 3.2 muestra el ejemplo 1.1 en el formato utilizado en Wikipedia, mientras que el cuadro 3.13 muestra el mismo ejemplo, pero en el formato XML utilizado en los corpus de la conferencia. En este caso, en el ejemplo de Wikipedia se utilizó la plantilla {*weasel-inline*}, que denota la existencia de *weasel words* en la oración. Este es el nivel de precisión más alto que provee Wikipedia para identificar *weasel words*. Notamos que en el

[2]http://www.inf.u-szeged.hu/rgai/conll2010st/

formato de la conferencia, se marcan exactamente cuáles son las *weasel words* de la oración mediante etiquetas específicas.

3.4. Diseño del experimento

De acuerdo con los objetivos del proyecto, separamos el experimento a realizar en dos fases. Primero, se construyen clasificadores tradicionales de aprendizaje supervisado con dos técnicas ampliamente utilizadas en el área: *Naive Bayes*, y Máquinas de Vectores de Soporte (*SVM*). En segundo lugar, una vez implementados estos modelos, se procede a los experimentos con *active learning*, tomando un clasificador basado en *Naive Bayes* como clasificador de base.

El corpus de entrenamiento se divide en dos subcorpus durante todo el desarrollo; uno de entrenamiento en sí (con 10000 oraciones), y otro de pruebas (con 1111 oraciones), sobre el cual se mide la performance de los modelos a medida que se van construyendo y mejorando. Los criterios de evaluación de los clasificadores coinciden con los tomados en la *Shared Task* de la ConLL 2010, y se presentan en la siguiente sección.

Es importante notar que, en los enfoques tradicionales, los ejemplos de entrenamiento son elegidos de manera aleatoria, mientras que en *active learning* los elige el algoritmo. Uno de los objetivos del experimento es precisamente comparar este aspecto de ambas metodologías, entrenando los clasificadores tradicionales con todo el corpus de entrenamiento por un lado, y permitiendo que el algoritmo de *active learning* elija sus ejemplos del mismo corpus en forma incremental por otro.

Las primeras pruebas realizadas involucran la implementación de un clasificador basado en *Naive Bayes*, con un conjunto de atributos básico. El clasificador se entrena sobre el subcorpus de entrenamiento y se calcula su performance sobre el subcorpus de pruebas. Luego, utilizando los mismos atributos, se implementa un clasificador basado en SVM, que es evaluado del mismo modo. Adicionalmente se lleva a cabo un conjunto de pruebas con el fin de ajustar los parámetros específicos de este modelo. Una vez implementados ambos modelos, se procede a la mejora del conjunto de atributos básico inicial. Como resultado de este proceso se obtiene un nuevo conjunto de atributos, con una serie de parámetros asociados, los cuales fueron ajustados a través de un nuevo conjunto de pruebas.

Finalizadas las pruebas con los modelos tradicionales, se realizan las primeras pruebas con *active learning*, utilizando como clasificador de base el clasificador con el cual se obtuvieron los mejores resultados sobre el corpus de pruebas (el clasificador basado en *Naive Bayes*, con los atributos mejora-

dos). Como se menciona antes, el objetivo de este experimento es comparar el desempeño de un mismo clasificador con la diferencia que aporta la selección dinámica de los ejemplos de entrenamiento. El algoritmo sería exitoso en caso de alcanzar niveles equivalentes de performance con una cantidad sustancialmente menor de ejemplos etiquetados que el total de ejemplos en el corpus de entrenamiento. De esta forma, nuestro experimento consiste en construir un marco de trabajo en el cual el algoritmo elige dinámica, inteligente e incrementalmente los ejemplos con los cuales entrenar el clasificador de base, midiendo la performance sobre el subcorpus de pruebas en cada paso.

En la sección que sigue se presentan los criterios de evaluación aplicados, para luego dar paso a la descripción de los clasificadores y las pruebas realizadas.

3.5. Criterios de evaluación

Con el fin de poder comparar los resultados, los criterios de evaluación utilizados en este trabajo coinciden con los adoptados en la *Shared Task* de la ConLL 2010. Dadas las predicciones de un clasificador y la clasificación correcta para un conjunto de ejemplos, se definen los siguientes subconjuntos:

- verdaderos positivos (VP): Oraciones inciertas correctamente etiquetadas por el clasificador.
- falsos positivos (FP): Oraciones factuales incorrectamente etiquetadas por el clasificador.
- verdaderos negativos (VN): Oraciones factuales correctamente etiquetadas por el clasificador.
- falsos negativos (FN): Oraciones inciertas incorrectamente etiquetadas por el clasificador.

A partir de estas definiciones se calculan la precisión, exhaustividad y su media armónica (F), como sigue:

- precisión (P): $\frac{VP}{VP+FP}$
- exhaustividad (R): $\frac{VP}{VP+FN}$
- media armónica (F): $2\frac{P*R}{P+R}$

La precisión mide la cantidad de aciertos con respecto al total de las predicciones de positivos, mientras que la exhaustividad mide la cantidad de oraciones inciertas reconocidas con respecto a su total. La media armónica es una medida resumen de amplio uso en el área.

```
#Extraccio'n de atributos para la oracio'n
# "The best of The Beatles"
{'the':2, 'be':0; 'of':1}
#Ejemplo de oracio'n etiquetada
({'the':2, 'be':0; 'of':1},'certain')
#Conjunto de entrenamiento con dos ejemplos
[({'the':2, 'be':0; 'of':1},'certain'),
({'the':1, 'be':1; 'of':1},'uncertain')]
```

Cuadro 3.3: Ejemplos de los formatos utilizados durante la implementación. Los atributos considerados se corresponden con las cantidades de ocurrencias de las palabras "the", "be" y "of" en la oración.

3.6. Clasificadores iniciales

Como se menciona en secciones anteriores, en primer lugar se trabajó en la construcción de un clasificador basado en *Naive Bayes*, y luego se realizó la implementación de un clasificador que utiliza SVM. En ambos casos se utilizó el mismo conjunto de atributos básico que toma en cuenta la cantidad de ocurrencias de las palabras en cada oración.

Nos centramos ahora en el proceso de selección, extracción y mejora de los atributos considerados. Detalles sobre la implementación de ambos modelos se presentan más adelante.

3.6.1. Extracción de atributos: Bag of Words

Una vez elegidos los modelos con los que trabajamos, es necesario definir la función de extracción de atributos para las oraciones. Esto es, llevar el texto de una oración a una representación uniforme como se muestra en el cuadro 3.3. En este primer acercamiento se eligieron como atributos las cantidades de ocurrencias de las palabras más frecuentes en el corpus de entrenamiento. Este enfoque se conoce como "bolsa de palabras" o "Bag of Words" (BOW), y fue uno de los modelos de base más ampliamente utilizado en la *Shared Task* de la ConLL 2010. En nuestro caso, el método consiste en contar los *tokens* del corpus de entrenamiento, y quedarse con los N más frecuentes, exceptuando los primeros M (que serán considerados como *stop words*) [3], que se considera no aportan información valiosa al modelo. Los valores que toman los atributos serán la cantidad de ocurrencias de cada uno de los *tokens* seleccionados en la oración en cuestión. Es importante destacar que

[3] $N > M$. En secciones siguientes se especifican los valores utilizados y su ajuste.

este enfoque es dinámico en sus atributos, ya que en la medida que cambie el conjunto de entrenamiento, cambian los atributos considerados.

3.6.2. Primeros resultados

En el cuadro 3.4 se presentan los primeros resultados obtenidos al evaluar el clasificador NB sobre el corpus de pruebas, con los atributos mencionados para $N = 1500$ y $M = 50$. En esta etapa se excluyeron los títulos de los artículos de ambos subcorpus (entrenamiento y de pruebas) porque se consideró que no aportaban información al modelo.

Clasificador	Precisión	Exhaustividad	Medida F
NB-BOW,N=1500,M=50	0,595	0,359	0,448

Cuadro 3.4: Primeros resultados con el clasificador NB-BOW sobre el subcorpus de pruebas.

El modelo basado en SVM admite diversos parámetros e específicos para construir el clasificador a partir del conjunto de entrenamiento. En particular se realizaron pruebas con los siguientes parámetros:

- Tipo de SVM: C-SVC, nu-SVC.
- Costo: Aplica cuando el tipo es C-SVC, y controla la relación entre los errores cometidos sobre los ejemplos de entrenamiento y la maximización de la distancia a los vectores de soporte. Puede variar en $[0, \infty]$.
- Nu: Aplica cuando el tipo es nu-SVC, y es el equivalente al parámetro anterior para este tipo de SVM. Puede variar en $[0, 1]$.

En el cuadro 3.5 se presenta un resumen de las pruebas realizadas combinando diferentes valores para estos parámetros. Se utilizó el conjunto de atributos implementado hasta el momento, es decir los atributos BOW, con $N = 1500$ y $M = 50$. Al igual que en la prueba anterior aquí también se excluyeron los títulos de los artículos. La combinación de parámetros que obtuvo mejores resultados en medida F es la que se utilizó en adelante para todos los clasificadores SVM.

3.7. Mejora de atributos

Luego de realizadas las primeras pruebas con ambos modelos de clasificadores se procedió a la mejora de los atributos. Hasta este punto y tal como

Tipo SVM	Costo	Nu	Precisión	Exhaustividad	Medida F
C-SVC	1	n/a	0,539	0,377	0,444
C-SVC	2	n/a	0,506	0,387	0,439
C-SVC	3	n/a	0,515	0,401	0,451
C-SVC	4	n/a	0,500	0,406	0,448
C-SVC	5	n/a	0,520	0,424	**0,467**
C-SVC	6	n/a	0,517	0,415	0,460
C-SVC	7	n/a	0,520	0,415	0,462
C-SVC	8	n/a	0,511	0,415	0,458
nu-SVC	n/a	0,1	0,206	**0,567**	0,302
nu-SVC	n/a	0,2	0,201	0,548	0,294
nu-SVC	n/a	0,3	0,405	0,493	0,445
nu-SVC	n/a	0,4	0,489	0,410	0,446
nu-SVC	n/a	0,5	**0,614**	0,323	0,423

Cuadro 3.5: Resumen de los resultados del ajuste de parámetros específicos para el clasificador SVM-BOW sobre el subcorpus de pruebas.

fuera mencionado en 3.6.1, se había empleado un enfoque basado en BOW simple para seleccionar los atributos de las oraciones. Este enfoque consistía en tomar los N *tokens* más frecuentes en el corpus de evaluación exceptuando a los primeros M, considerados como *stop words*. Posteriormente, se agregaron las siguientes mejoras:

- Bag of Lemmas (BOL): Enfoque análogo al anterior BOW, donde se sustituyen los *tokens* por sus correspondientes lemas.
- Entidades con nombre propio (NEs): Se agrupan en un *token* genérico aquellos *tokens* que juntos representan un nombre propio.
- Expresiones regulares para *weasel words* (Wexes): Se confecciona una lista de expresiones regulares para detectar expresiones utilizadas frecuentemente en oraciones inciertas.

En las siguientes secciones se describe la implementación de estas mejoras.

3.7.1. Bag of Lemmas y entidades con nombre

El primer cambio introducido tiene como objetivo mejorar el proceso de generalización del algoritmo, llevando cada palabra a su forma canónica.

```
#Extraccio'n de atributos (BOL)
#"The authenticity of the manuscript has been challenged,
#   while some say it can't possibly be fake."
{'the':2, 'be':2; 'of':1}
```

Cuadro 3.6: Ejemplo de extracción de atributos con el agregado de lemas.

```
#Extraccio'n de atributos (BOL+NES)
#"The best of The Beatles."
{'the':1, 'be':0; 'of':1; 'NES':1}
```

Cuadro 3.7: Ejemplo de extracción de atributos con el agregado de lemas y reconocimiento de entidades con nombre.

Esto es, diferentes conjugaciones de un mismo verbo, las formas singulares y plurales de un mismo sustantivo, o las diferentes formas comparativas de adjetivos se agrupan en un mismo atributo, en vez de tomar un atributo por cada una. De forma análoga al enfoque BOW, ahora los atributos pasan a ser los lemas más frecuentes en el corpus de entrenamiento, exceptuando los primeros. En el cuadro 3.6 se ejemplifica la nueva función de extracción de atributos.

En la misma línea, se agrega un atributo que cuenta la cantidad de entidades con nombre en la oración. En consecuencia, las palabras que son parte de una entidad con nombre dejan de ser consideradas para los atributos del BOL y se agrupan en un solo objeto que suma a la cuenta del nuevo atributo. En el cuadro 3.7 se muestra un ejemplo de la función de extracción de atributos que incluye el nuevo atributo.

Ambos cambios contribuyen a la generalización en cuanto a que se reduce considerablemente el tamaño del conjunto de atributos posibles, ya que la cantidad de lemas distintos es menor que la cantidad de palabras distintas.

3.7.2. Wexes

Por último, se agregó una serie de atributos que se corresponden con una lista de expresiones regulares que intentan capturar información relevante para distinguir oraciones inciertas. Llamamos a estas expresiones Wexes (acrónimo plural de *"weasel expression"*). La lista fue confeccionada en forma manual a partir de una serie de expresiones recolectadas de Wikipedia, y posteriormente extendida y refinada a partir del análisis de errores en el subcorpus de pruebas. Luego se extendió cada frase de la lista mediante una

expresión regular de forma de expandir su alcance, intentando mantener las características relevantes para distinguir las *weasel words*. En el cuadro 3.8 se muestra la lista utilizada en las pruebas.

Se agregó un atributo binario por cada expresión, y se modificó la función de extracción de atributos para que intente emparejar cada expresión con la oración en cuestión, dando el valor 1 al atributo correspondiente en casos de emparejamiento positivo o 0 en caso contrario. En el cuadro 3.9 se presenta un ejemplo de la función final extracción de atributos. Más adelante, al presentar los resultados, veremos que el agregado de estos atributos genera una mejora significativa en la performance de los clasificadores.

3.8. Ajuste de parámetros

Una vez implementados los clasificadores y luego de realizada la mejora de los atributos, podemos seguir con el ajuste de parámetros. Al igual que en pruebas anteriores, se divide el corpus de entrenamiento en dos subcorpus, uno de entrenamiento en sí, y otro de pruebas, sobre el cual se evalúan los clasificadores. En particular, los parámetros que variaron en este experimento son:

1. Utilización de BOW o BOL.
2. Los parámetros N y M del BOW (y BOL), es decir la cantidad de palabras que se consideran dentro de las más frecuentes (N) y cuáles de esas son las que se descartan (las primeras M).
3. Utilizar atributos con dominio natural o booleano en el BOW (o BOL). Es decir, contar las ocurrencias de cada *token*, o utilizar atributos binarios representando la ocurrencia o no del *token* en la oración.

Donde:

$$N \in \{5000, 4000, 3000, 2000, 1000, 500\} \text{ y } M \in \{50, 20, 10, 0\}$$

El reconocimiento de nombres propios se mantuvo en todos los casos. El ajuste se hizo sobre ambos modelos (*Naive Bayes* y SVM), y se ejecutaron pruebas con todas las combinaciones de valores para los parámetros mencionados, resultando en un total de 96 casos para cada modelo. Luego se tomó la mejor configuración de valores de cada modelo y se agregó la lista de wexes. En el cuadro 3.10 se resumen los resultados. A primera vista llama la atención que en ambos casos sea mejor utilizar la variante en la que se utilizan atributos binarios para el BOL. Es posible que este comportamiento

```
'it\s+(has\s+been|was|is)\s+.*(said|claimed|noted|stated|indicated|
    argued| believed|mentioned|perceived|expected|commented|thought|
    speculated|shown)',
'it\s+(is|has|was)\s+.*ed'
'(some|many|most)\s+people'
'(some|many|most)\s+people\s+(have\s+)?'
'(some|many|most)\s+(have\s+)?((people)\s+)?(\w*d|known|thought)'
'(some|many|most)\s+((people)\s+)?(say|claim|note|state|indicate|
    argue|believe|mention|perceive|expect|comment|think|speculate|
    show)'
'(studies|experience)\s+.*(say|claim|note|state|indicate|argue|
    believe|mention|perceive|expect|comment|think|speculate|show)'
'according\s+to\s+((some|others|many|most|(the\s+majority)|(the\s+
    minority)|several|(a\s+wide)|various|few|numerous)\s+)?(people\s
    +)?'
'some|people|others|many|most|(the\s+majority)|(the\s+minority)|
    several|(a\s+wide)|various|few|numerous'
'(more\s+than|up\s+to|upwards\s+of)'
'mostly|generally|usually|commonly|greatly|vastly|mainly|relatively
    |poorly|universally|widely'
'(has|have)\s+been'
'(are|is|was|were)\s+(\w*d|known|thought)'
'(i\s+heard\s+)|(it\s+(stands\s+to\s+reason|is\s+known|turns\s+out|
    was\s+noted))|(there\s+is\s+(no\s+)?evidence\s+)|appears|seems|(
    looks\s+like)|(is\s+(most\s+)?likely)'
'almost'
'might|may'
'responsible'
'principal|primary|main'
'better|significant|important|great|extensive|widespread|vast'
'apparently|likely|virtually'
'supposedly|allegedly|seemingly'
'would'
'known'
'controversial'
'increasing'
'important'
'critics'
'clearly'
'claim'
'series\s+of'
```

Cuadro 3.8: La lista completa de Wexes utilizada.

```
#Extraccio'n de atributos (BOL+NES+WEXES)
oracion = "The authenticity of the manuscript has been challenged,
    while some say it can't possibly be fake."
WX1 = 'it\s+(has\s+been|was|is)\s+.*(said|claimed|...|shown)',
WX2 = '(some|many|most)\s+((people)\s+)?(say|claim|...|show)'
WX3 = 'mostly|generally|...|widely'
WX4 = '(has|have)\s+been'
fset = {'the':2, 'be':2; 'of':1, 'NES': 0, 'WX1': 0, 'WX2':1 , 'WX3
    ':0, 'WX4':1}
```

Cuadro 3.9: Ejemplo de extracción de atributos que incorpora todas las mejoras. Por razones de claridad, sólo se utilizan cuatro wexes.

Parámetros	Modelo	Precisión	Exhaustividad	Medida F
BOL,CB,N=500,M=10	Naive Bayes	0,589	0,544	0,566
BOL,CB,N=500,M=10,WEXES	Naive Bayes	0,653	**0,595**	**0,623**
BOL,CB,N=500,M=0	SVM	0,787	0,443	0,567
BOL,CB,N=500,M=0,WEXES	SVM	**0,800**	0,456	0,581

Cuadro 3.10: Resumen de los resultados del ajuste de parámetros (sobre el corpus de pruebas). BOL: Bag of lemmas, CB: Conteo booleano.

se dé porque al contar, los valores de los atributos resultan un poco más dispersos, mientras que utilizando solo dos valores los ejemplos son más fáciles de agrupar.

Además, se corrobora que es mejor (en el corpus de pruebas) utilizar los lemas de los *tokens* en vez de los *tokens* mismos. Observamos una mejora sustancial en ambos casos al agregar la lista de expresiones de *weasels*, donde además el clasificador NB supera en aproximadamente 4 puntos al clasificador SVM. Por otro lado, el clasificador SVM logra la mayor precisión aunque baja exhaustividad, mientras que el clasificador NB tiene valores menos separados de precisión y exhaustividad que le permiten alcanzar una mejor performance en medida F.

Con estas pruebas se concluye el trabajo sobre los clasificadores tradicionales, y se da paso a las pruebas con *active learning*.

3.9. Active Learning

En esta sección se describe el enfoque tomado para la aplicación de métodos de *active learning* al problema de la detección de oraciones inciertas. Recordemos que el objetivo básico del *active learning* es minimizar la canti-

dad de consultas a oráculos, eligiendo las consultas de forma conveniente. La intención de este experimento es estudiar la aplicabilidad de técnicas básicas de *active learning* esperando alcanzar los resultados anteriormente mencionados.

De los tres escenarios mencionados en Settles (2009), el más conveniente resulta ser aquel en que se tiene acceso a los ejemplos sin etiquetar, para elegir de entre ellos los más adecuados para enviar al oráculo. Uno de los otros escenarios posibles implica la síntesis de oraciones, que escaparía a los alcances de este trabajo y el restante implica tener un flujo de oraciones de ejemplo, donde los ejemplos van apareciendo de a uno por vez y el algoritmo decide en el momento si consultar por su etiqueta o no. Si bien este caso se pudo haber simulado con el corpus de entrenamiento, se decidió implementar el caso mencionado anteriormente, bajo el supuesto de que es mejor poder analizar todos los ejemplos antes de seleccionar uno para consultar.

Por otro lado, una de las técnicas de selección de consultas más utilizadas es la selección por incertidumbre, en la que se elige aquel ejemplo que minimiza la confianza que el clasificador tiene en su predicción. Llevado a nuestro caso, el conjunto de ejemplos sin etiquetar será el corpus de entrenamiento (simulando que se desconocen sus etiquetas), y debido a que estamos realizando clasificación binaria, nuestra estrategia consistirá en seleccionar para consultar aquella oración cuya etiqueta más probable tenga la probabilidad más próxima a 0,5.

El oráculo se simula con el mismo corpus anotado. Esto es, en vez de tener un experto que clasifique los ejemplos elegidos para consultar su etiqueta, simplemente se devuelve la etiqueta asignada en el corpus. De esta forma podemos comparar los resultados en cantidad de consultas y performance contra el enfoque tradicional.

Definidos el escenario y el método de selección de consultas, es importante notar que el planteo del *active learning* es independiente del modelo subyacente. Respetando esta noción, se implementó el algoritmo de forma de poder desacoplar la metodología del clasificador de base. En el cuadro 3.11 se muestra el pseudocódigo del algoritmo implementado. Podemos ver que el único requisito para el clasificador de base es tener la capacidad de entrenarse a partir de un conjunto inicial de ejemplos etiquetados, y luego poder interiorizar nuevos ejemplos de forma incremental, a medida que se van consultando. En nuestro caso, se implementó la versión activa del clasificador basado en *Naive Bayes*, y fue necesario realizar algunos cambios en la implementación para cumplir con este requisito.

```
conjunto_entrenamiento = conjunto_inicial
clasificador = ENTRENAR conjunto_entrenamiento
consultas = CONTAR conjunto_inicial
MIENTRAS (performance < umbral Y NO VACIO pool)
       oracion_incert = BUSCAR_MIN_PROB pool
       BORRAR oracion_incert DE pool
       ACTUALIZAR clasificador CON oracion_incert
       performance = EVALUAR clasificador EN corpus_prueba
       consultas++
FIN MIENTRAS
DEVOLVER clasificador, consultas
```

Cuadro 3.11: Pseudocódigo del algoritmo de active learning aplicado.

Naive Bayes incremental

La forma más directa de lograr una versión incremental del algoritmo de *Naive Bayes* es reentrenar el clasificador cada vez que llega un nuevo ejemplo. Internamente, esto implica volver a hacer recuentos de ocurrencias de etiquetas y valores de atributos que ya son conocidos. Para lograr el mismo resultado y evitar hacer recuentos innecesarios se implementó una versión incremental del algoritmo anterior. En particular, al entrenar un clasificador de este tipo es necesario calcular:

- La distribución de frecuencias de las etiquetas, a partir de la cantidad de ejemplos en cada categoría (dos en nuestro caso).
- Para cada atributo y para cada categoría, la distribución de frecuencias de los valores del atributo. Por ejemplo, para un atributo a con dominio $\{0, 1\}$, en oraciones inciertas, se calcula a partir de la cantidad de veces que el atributo toma cada uno de los valores en este tipo de oraciones.

A partir de estas distribuciones se crean estimadores de probabilidad de "verosimilitud esperada" (del inglés *"Expected likelihood estimate"*), que son equivalentes a los de máxima verosimilitud donde previamente se suma 0,5 a cada valor observado. Finalmente con los estimadores instanciados se crea el clasificador. Entonces para implementar el modelo incremental, basta con mantener los contadores mencionados, actualizarlos cuando llega un nuevo ejemplo (incrementar los campos indicados) y volver a crear los estimadores. En el cuadro 3.12 se muestra la estructura general del algoritmo incremental.

Hasta este punto, tenemos un algoritmo que en cada iteración recorre y estima la probabilidad de contener *weasels* para cada oración en el conjunto

```
PROC actualizarClasificador(clasificador,ejemplo)
  netq = ejemplo.etiqueta
  natrs = ejemplo.atributos_valores
  #actualiza la cantidad de ejemplos etiquetados con netq
  INCREMENTAR clasificador.frecdist_etiquetas, netq
  PARA CADA atr,val EN natrs:
    #actualiza frec(val|netq, atr)
    INCREMENTAR clasificador.frecuencias_atr[atr,netq], val
  #crea la distribucio'n para las etiquetas
  FIN PARA
  distr_etiquetas = ESTIMADOR(clasificador.frecuencias_etq)
  #contiene la distribucio'n para los valores de los atributos para
      cada etiqueta
  distrs_atributos = []
  PARA CADA ((etq, atr), frecdist) EN clasificador.frecuencias_atr:
    #crea la distribucio'n para estimar P(valor|etq,atr)
    #frecdist es la distribucio'n por frecuencias
    #de valores para el atributo atr y la etiqueta etq
    probdist = ESTIMADOR(frecdist)
    distrs_atributos[etq,atr] = probdist
  FIN PARA
  clasificador = CREAR_CLASIFICADOR distr_etiquetas,
      distrs_atributos
FIN PROC
```

Cuadro 3.12: Pseudocódigo del algoritmo de Naive Bayes incremental. Se parte de un clasificador ya entrenado y un nuevo ejemplo etiquetado, y se devuelve el mismo clasificador actualizado con la información del nuevo ejemplo.

de oraciones sin etiquetar. En vista de los altos costos computacionales que esto conlleva, se implementó una variante del algoritmo anterior que selecciona pequeños subconjuntos en cada iteración en vez de tomar los ejemplos de a uno. Esta última versión es la que se utilizó finalmente en todas las pruebas con *active learning*.

3.10. Detalles de implementación

En esta sección se presentan algunos detalles de implementación de los clasificadores construidos. El lenguaje elegido para el desarrollo del proyecto fue Python debido a su facilidad de uso y mantenimiento.

3.10.1. Naive Bayes

Para construir los clasificadores basados en *Naive Bayes* se utilizó el *Natural Language Toolkit* (NLTK)[4] de Python. Se trata de una biblioteca extensamente utilizada en el área, que ofrece herramientas simples y eficientes para el procesamiento de lenguaje natural. En particular, el módulo empleado define un formato simple para los ejemplos de entrada, y para entrenar un clasificador basta con proveer un conjunto de ellos. Completado el entrenamiento podemos etiquetar nuevos ejemplos y, en consecuencia, evaluar la performance del clasificador que se obtiene. En lo que sigue, se describe la serie de transformaciones necesarias para utilizar la clase `NaiveBayesClassifier` de NLTK a partir de los corpus de la *Shared Task* de la ConLL 2010.

Formato de entrada de NLTK

Al momento de la clasificación, un ejemplo queda representado por los valores que toman sus atributos. Con este fin, en NLTK se utilizan arreglos asociativos (diccionarios), que asocian un valor a cada atributo. Las claves del diccionario se corresponden con los nombres de los atributos, y los valores del atributo están dados por los valores asociados a la respectiva clave en el diccionario. Un ejemplo etiquetado se representa como un par ordenado con los atributos del ejemplo y la etiqueta correspondiente. Finalmente, un conjunto de ejemplos es simplemente una lista de ejemplos etiquetados. En el cuadro 3.3 se muestran algunos ejemplos en sintaxis Python.

[4] http://nltk.org/

```
<sentence certainty="uncertain" id="S1.2">
       In children, this type of hearing loss can lead to
       social isolation for <ccue>several reasons</ccue>.
</sentence>
```

Cuadro 3.13: Ejemplo de una oración en el formato provisto en la Shared Task de la ConLL 2010.

```
{id:'S1.2',
certainty:'uncertain',
string:'In children, this type of hearing loss can lead to social
    isolation for several reasons.'}
```

Cuadro 3.14: La representación en formato Python (un objeto de tipo `Sentence`) de la oración del ejemplo 3.13.

Lectura de los corpus de *Shared Task* de la ConLL 2010

Disponiendo del corpus etiquetado y de la especificación del formato de entrada, fue preciso contar con una representación intermedia que permitiera el fácil manejo de los ejemplos durante el desarrollo del trabajo. Con este propósito, se definió un conjunto de clases Python que se encargan de crear objetos a partir de texto XML, mediante la biblioteca `xml.dom.minidom`[5]. En los cuadros 3.13 y 3.14 se muestra un ejemplo de conversión de texto XML a un objeto de tipo `Sentence`.

Para obtener los lemas de cada palabra y reconocer nombres propios se utilizó el *tagger* GENIA[6]. Esta herramienta es externa a Python, por lo cual resultó necesario hacer uso de la biblioteca `subprocess` que permite ejecutar procesos externos. Se desarrolló un pequeño script de preprocesamiento que permite cargar en las oraciones la información de cada *token* que provee GENIA. En el cuadro 3.15 se muestra un ejemplo de salida del *tagger*. El procedimiento de carga de los nuevos atributos consiste en escribir un archivo con una oración por línea, que será entrada para GENIA. De acuerdo al formato de salida de GENIA se cargan en cada objeto `Sentence` los datos para cada uno de los *tokens*. Al momento de extraer los valores para las oraciones, la función utiliza los lemas de cada *token* para actualizar el diccionario de atributos y excluye aquellos *tokens* que son parte de un nombre propio. Finalmente se agrega un atributo al diccionario que representa la cantidad

[5] http://docs.python.org/2/library/xml.dom.minidom.html
[6] http://www.nactem.ac.uk/GENIA/tagger/

Palabra	Lema	POS	CHUNK	NE
The	The	DT	B-NP	O
best	best	NNP	I-NP	O
of	of	IN	B-PP	O
The	The	DT	B-NP	B
Beatles	Beatles	NNPS	I-NP	I
.	.	.	O	O

Cuadro 3.15: Ejemplo de salida de GENIA.

```
#Diccionario en NLTK
{(atributo1,valor1),(atributo2,valor3),(atributo3,valor3)}
#Secuencia en libsvm
[valor1,valor2,valor3]
```

Cuadro 3.16: Ejemplo del formato para las oraciones requerido por `libsvm`.

de nombres propios que aparecen en la oración.

3.10.2. SVM

Para construir los clasificadores basados en SVM se recurrió a la biblioteca para Python provista por `libsvm` (Chang and Lin, 2011). Debido a algunas diferencias en los formatos entre `libsvm` y NLTK se hicieron modificaciones en la función de extracción de atributos de forma de que sea posible utilizarla en ambos modelos.

En `libsvm` se definen *problemas* a partir de los cuales se entrenan los clasificadores. Cada problema se crea a partir de una lista de ejemplos de oraciones representadas por sus atributos, y una lista de igual largo con las etiquetas correspondientes a cada oración. Asumiendo que se tienen valores para todos los atributos definidos para las oraciones, la biblioteca exige que se represente cada oración como la secuencia de valores para los atributos elegidos. En nuestro caso esto se cumple por la forma en que se extraen los atributos. Además, tanto las etiquetas como los valores que toman los atributos deben ser llevados a tipos numéricos. En 3.16 se muestra un ejemplo de conversión del formato de NLTK al requerido por `libsvm`.

Capítulo 4

Evaluación

Dedicamos este capítulo a la presentación de los resultados que se obtuvieron al evaluar los clasificadores implementados. Con fines experimentales se realizaron pruebas con un conjunto de atributos básico (BOW) y con el conjunto de atributos mejorados (que llamamos BOL). Ambas pruebas fueron llevadas a cabo con los dos enfoques tradicionales (*Naive Bayes* y SVM) utilizando todo el corpus de entrenamiento. Por otro lado, para realizar las pruebas con *active learning* sobre el clasificador basado en *Naive Bayes*, se separó el corpus de entrenamiento en dos subcorpus, uno de entrenamiento, y otro de pruebas.

La performance se calculó en términos de precisión, exhaustividad y su media armónica (medida F), en función de la cantidad de ejemplos de entrenamiento.

A continuación, se muestran en primer lugar los datos correspondientes a la performance sobre el corpus de evaluación de los clasificadores tradicionales. Luego se continúa con la presentación y el análisis de los resultados del experimento con *active learning* y finalmente se cierra el capítulo con algunos comentarios sobre los errores más comunes cometidos por los clasificadores.

4.1. Naive Bayes y SVM

Las figuras 4.1 y 4.2 muestran las curvas de aprendizaje que se obtuvieron con el conjunto de atributos básico (BOW) para los clasificadores NB y SVM, respectivamente. El eje horizontal representa la cantidad de ejemplos del corpus de entrenamiento (con un máximo de 11111 oraciones) y el eje vertical mide la performance (precisión, exhaustividad (*recall*) y medida F) sobre el corpus de evaluación. El tamaño del corpus de evaluación es de 9634 oraciones. Esto permite visualizar la evolución de la performance en

función de la cantidad de ejemplos de entrenamiento. Resultados análogos se presentan en las figuras 4.3 y 4.4 para el conjunto de atributos mejorados.

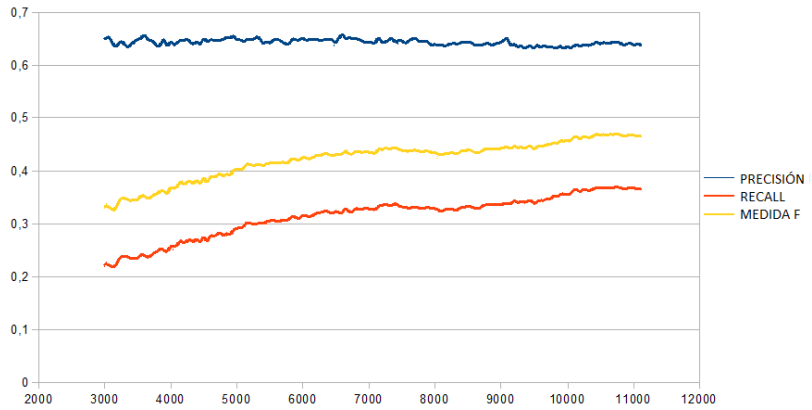

Figura 4.1: Evolución de los resultados sobre el corpus de evaluación, para el clasificador BOW-NB.

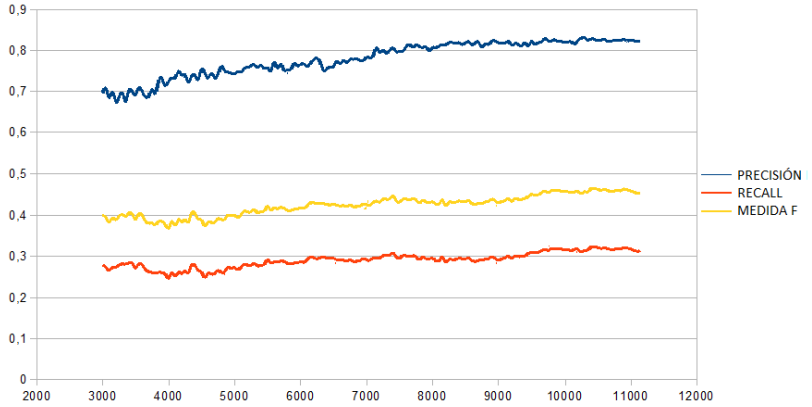

Figura 4.2: Evolución de los resultados sobre el corpus de evaluación, para el clasificador BOW-SVM.

La tabla 4.1 presenta información resumida de las pruebas anteriores. En particular se muestra el punto (la cantidad de consultas) en que se alcanzaron los máximos en medida F. Los mejores resultados en medida F son alcanzados por los clasificadores NB con una puntuación de 0,523, mientras que los clasificadores SVM llegan sólo a 0,467. Podemos considerar que estos resultados son razonables comparados con los resultados de la *Shared Task* de la ConLL 2010, donde el mejor resultado fue de 0,602 en medida F (0,720 y 0,517 en precisión y exhaustividad respectivamente).

Al analizar las medidas, observamos que la precisión alcanzada por SVM es varios puntos mayor a la de NB (0,835 sobre 0,689 de NB), relación que se

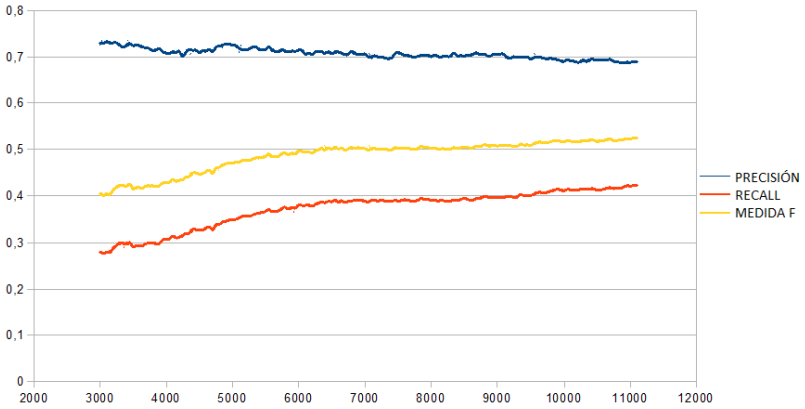

Figura 4.3: Evolución de los resultados sobre el corpus de evaluación, para el clasificador BOL-NB.

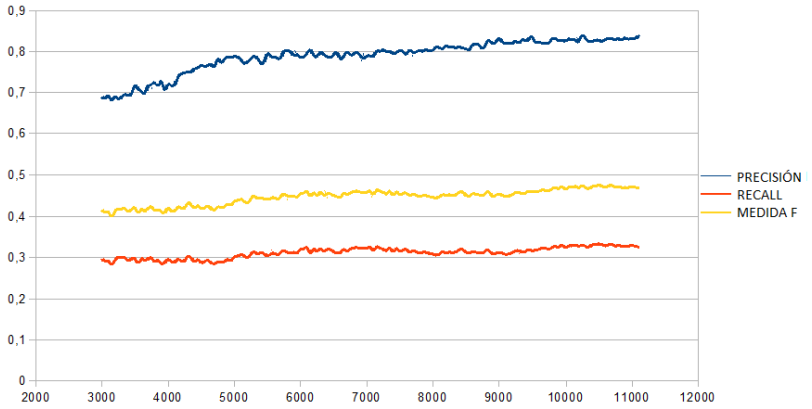

Figura 4.4: Evolución de los resultados sobre el corpus de evaluación, para el clasificador BOL-SVM.

invierte al observar la performance en exhaustividad, donde NB alcanza 0,422 mientras que SVM logra una puntuación bastante más baja de 0,33. Estos resultados sugieren que los clasificadores NB sacrifican puntos de precisión arriesgando más predicciones positivas, dando lugar a una mayor exhaustividad, mientras que los clasificadores SVM se mantienen más seguros, logrando una precisión mayor pero con un cubrimiento menor de las *weasel words* que aparecen en el corpus de evaluación.

En todos los casos, si bien los máximos se dan antes de utilizar todo el corpus de entrenamiento, las diferencias son menores con respecto a la performance alcanzada utilizándolo completamente, por lo que es factible decir que al aumentar el tamaño de corpus, aumenta la performance. Considerando además que no se observa un estancamiento en las curvas, resultaría

interesante poder realizar nuevas pruebas aumentando aún más el tamaño del corpus de entrenamiento.

Notamos también en todos los casos una diferencia considerable entre los resultados en precisión y exhaustividad, siendo la primera varios puntos superior a la segunda. Estos resultados son frecuentes en la literatura, donde en general se concluye que la mayor dificultad en la tarea de detección de especulaciones es la de detectar aquellas especulaciones que no ocurren en el corpus de entrenamiento.

Finalmente notamos que el refinamiento de los atributos produce una mejora en la performance de manera similar a lo ocurrido durante la etapa de pruebas, aunque el efecto no es tan significativo en este caso posiblemente por sobreajuste al corpus de entrenamiento.

Avanzamos a la siguiente sección con la presentación de los resultados de las pruebas con *active learning*, tomando como base el clasificador BOL-NB.

Clasificador	#ejemplos	Precisión	Exhaustividad	medida F
BOW-NB	10750	0,642	0,369	0,469
BOW-NB	11111	0,635	0,363	0,462
BOW-SVM	10400	0,827	0,321	0,463
BOW-SVM	11111	0,823	0,313	0,454
BOL-NB	11100	0,689	**0,422**	**0,523**
BOL-NB	11111	0,687	0,421	0,522
BOL-SVM	10500	0,827	0,332	0,474
BOL-SVM	11111	**0,835**	0,325	0,467

Cuadro 4.1: Resumen de performance para los clasificadores tradicionales. Se muestran los puntos en que se maximiza la medida F, y los resultados obtenidos al utilizar el corpus de entrenamiento en su totalidad.

4.2. Active Learning

Luego de la ejecución con los enfoques tradicionales se pasó a evaluar la versión activa por lotes del clasificador BOL-NB. La idea de esta prueba es, entrenar el clasificador de base con un pequeño conjunto inicial, y luego permitir que el algoritmo de *active learning* elija los siguientes ejemplos para añadir a su conjunto de etiquetas conocidas. En cada paso se evalúa la performance sobre un subconjunto fijo del corpus de entrenamiento, y el proceso termina al alcanzar un umbral preestablecido. Una vez finalizada la

fase de entrenamiento, se pasa a evaluar el clasificador resultante (entrenado con los ejemplos elegidos interactivamente) sobre el corpus de evaluación. De esta forma podemos observar la diferencia entre tomar los ejemplos secuencialmente, o elegirlos mediante selección por incertidumbre.

Para estas pruebas se realizó la siguiente división del corpus de entrenamiento (con 11111 oraciones):

- corpus de consulta: un conjunto de 10000 oraciones de las cuales se seleccionan las consultas
- corpus de prueba: un conjunto de 1111 oraciones utilizadas para evaluar el clasificador en cada paso

El clasificador de base utilizado es el BOL-NB, que utiliza *Naive Bayes* y el conjunto de atributos mejorados. La performance de este clasificador sobre el corpus de pruebas se presentó en el cuadro 3.10.

Para comenzar la ejecución se tomaron las primeras 3000 oraciones del corpus de consulta para entrenar el clasificador inicial. Basado en sus estimaciones de probabilidades de cada etiqueta, este clasificador elige un conjunto de ejemplos para consulta y lo interioriza, resultando en un nuevo clasificador, y así sucesivamente.

En las pruebas, el algoritmo seleccionó en cada iteración conjuntos de 50 ejemplos para enviar a consulta. El proceso termina cuando la performance en el corpus de prueba supera un umbral preestablecido, o cuando se agota el corpus de consulta. Este umbral no debiera de superar la performance del clasificador de base, porque en realidad se trata del mismo clasificador donde simplemente se cambia el orden en que se toman los ejemplos de entrada. En este caso el valor del umbral de performance elegido fue de 0,6 en medida F, dado que el valor máximo para esa medida obtenido por el clasificador de base es de 0,6 sobre el corpus de pruebas.

Adicionalmente se realizó una prueba con un umbral mayor, para consumir todos los ejemplos y poder graficar la evolución de performance sobre el corpus de prueba (figura 4.6).

Los resultados de la primera prueba se muestran en los cuadros 4.2 y 4.3. El umbral de performance es alcanzado al cabo de 7172 consultas.

Clasificador	#consultas	Precisión	Exhaustividad	medida F
BOL-AL	7172	0,815	0,475	0,600

Cuadro 4.2: Performance sobre el corpus de prueba de la versión activa del clasificador NB, al cabo de 7172 consultas.

Clasificador	#consultas	Precisión	Exhaustividad	medida F
BOL-AL	7172	0,799	0,321	0,458

Cuadro 4.3: Performance sobre el corpus de evaluación de la versión activa del clasificador NB, al cabo de 7172 consultas.

Observamos que la performance en términos de precisión se mantiene. No pasa lo análogo con la exhaustividad, que sufre una baja de 0,16, dando lugar a una baja en los resultados para la medida F.

Para poder comparar mejor ambos enfoques (*active learning* y entrenamiento tradicional), se graficó también la evolución de la performance del clasificador NB tradicional sobre el corpus de prueba, en función del tamaño del corpus de entrenamiento (a medida que se agregan secuencialmente los ejemplos del corpus de consulta). Adicionalmente, se aseguró de que este clasificador considerara todas las palabras del corpus de consulta para armar su conjunto de atributos, en particular los atributos dados por el Bag of Lemmas, ya que de otra forma estos cambiarían en cada paso, y no es el comportamiento que toma la versión activa, que también considera todas las palabras al inicio. Estos resultados se muestran en la figura 4.5.

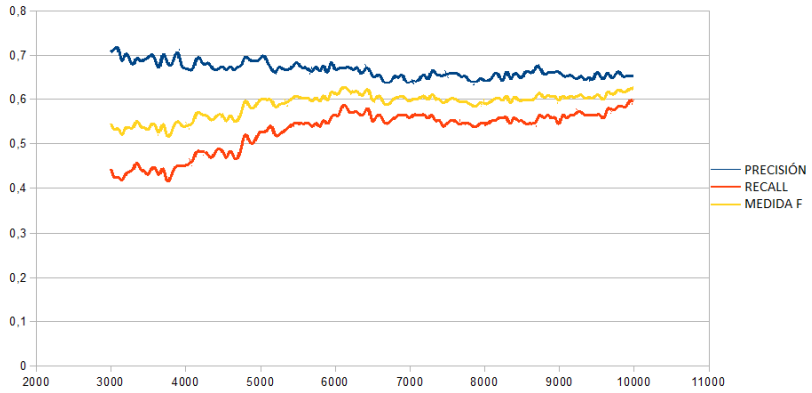

Figura 4.5: Evolución de los resultados sobre el corpus de prueba, para el clasificador BOL-NB, con atributos fijos.

La figura 4.7 permite visualizar conjuntamente la evolución de las tres medidas de evaluación sobre el corpus de prueba en función de la cantidad de ejemplos consultados del corpus de entrenamiento, para las versiones tradicional y activa del clasificador NB. Los ejemplos elegidos por el algoritmo de *active learning* parecen actuar en favor de la precisión, dejando de lado la mejora de la exhaustividad. Esto tiene sentido si se tiene en cuenta la dificultad del clasificador para identificar nuevas *weasel words*, que proba-

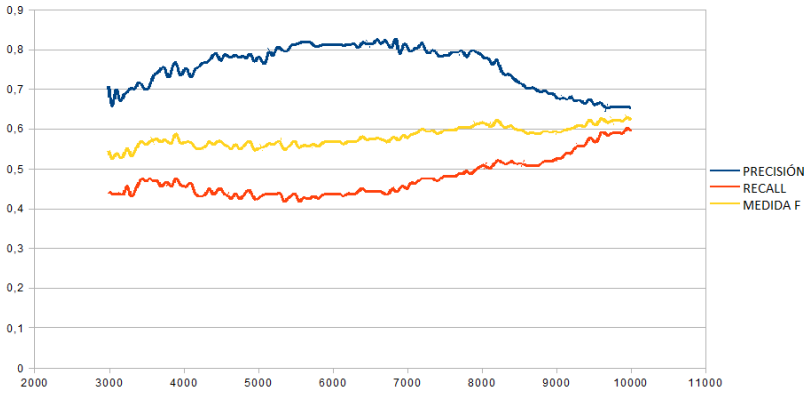

Figura 4.6: Evolución de performance sobre el corpus de prueba, para el clasificador BOL-AL. El umbral de 0,6 en medida F se alcanza al cabo de 7172 consultas.

blemente resultarán falsos negativos en la evaluación final. Es probable que exista un sesgo en las consultas realizadas hacia aquellos ejemplos que contienen *weasel words* conocidas en un contexto en que el algoritmo no termina de decidir con suficiente seguridad la etiqueta adecuada. En el análisis de errores a continuación se muestran algunos de estos ejemplos. En cuanto a la medida F, observamos que el orden en que el algoritmo elige los ejemplos de entrenamiento no parece acelerar su crecimiento, presentando una curva de aprendizaje similar a la obtenida agregando los ejemplos secuencialmente.

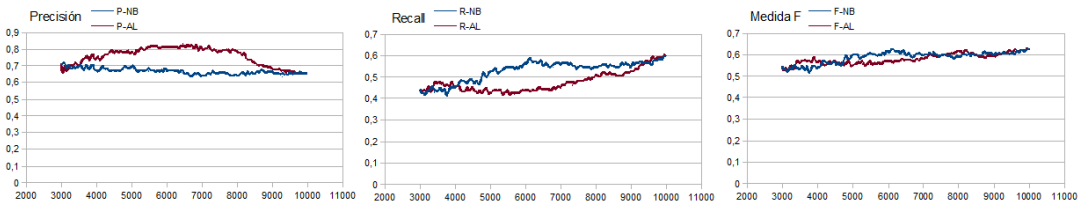

Figura 4.7: Gráficos conjuntos de evolución de precisión, exhaustividad y medida F sobre el corpus de prueba para las versiones activa y tradicional del clasificador NB.

4.3. Análisis de errores

Procedemos al análisis cualitativo de los errores cometidos por el clasificador BOL-NB. Al revisar los resultados, observamos que la performance de éste y el resto de los clasificadores implementados se ve severamente afectada por la baja performance en exhaustividad, en relación directa con la alta

Clasificador	Verdaderos positivos	falsos positivos	Verdaderos negativos	falsos negativos
BOL-NB	940	428	6972	1294
BOL-SVM	725	143	7257	1509

Cuadro 4.4: Resultados de las predicciones de los clasificadores BOL.

cantidad de falsos negativos en las predicciones. El cuadro 4.4 detalla los resultados de las predicciones de los clasificadores con los atributos mejorados. Una posible causa de estos errores queda expuesta al inspeccionar las *weasel words* que aparecen en el corpus de evaluación, en donde se observa un total de 2137 frases distintas anotadas como *weasel words*, de las cuales el 86 % no aparece en el corpus de entrenamiento, el 11 % aparece entre una y cinco veces, y sólo el 2 % restante aparece seis o más veces. Es decir, una proporción importante de las *weasel words* son desconocidas para el clasificador al momento de la evaluación, lo cual probablemente provoca la mayoría de los falsos negativos.

Al examinar los falsos positivos encontramos diferentes causas de error. Por un lado, están aquellos ejemplos que contienen varias expresiones que coinciden con wexes. Un ejemplo de esto es la oración en el ejemplo 4.1 que contiene tres expresiones que son comúnmente utilizadas en oraciones con *weasel words* (*"It was used"*, *"critics"* y *"was adopted"*).

Ejemplo 4.1 (falsos positivos - Wexes)

- *It was initially used solely by critics, such as Mirabeau and Smith, but was quickly adopted by historians. ("Era inicialmente utilizado únicamente por críticos, como Mirabeu y Smith, pero fue rápidamente adoptado por historiadores.")*

Se encontraron también casos como el del ejemplo 4.2 en el cual el clasificador encuentra una proposición que aparenta ser incierta, pero que es seguida de un listado explícito de sustantivos que atestiguan lo anteriormente mencionado. Es probable que este tipo de errores pueda evitarse mediante la mejora en los atributos que conciernen a las entidades con nombre.

Ejemplo 4.2 (falsos positivos - Listado explícito)

- *Pinchot has appeared in several feature films, including 'Risky Business', 'Beverly Hills Cop' , 'True Romance', 'Courage Under Fire' and 'It's My Party'. ("Pinchot ha participado en varios largometrajes, incluyendo 'Risky Business', 'Beverly Hills Cop' , 'True Romance', 'Courage Under Fire' y 'It's My Party'.")*

Hay casos como el del ejemplo 4.3 en que la semántica es importante para decidir la etiqueta correcta. En este caso, el universo de discurso permite asegurar que la oración es factual, pero existen oraciones sintácticamente equivalentes que son inciertas.

Ejemplo 4.3 (falsos positivos - Semántica)

- *Just before entering the valley it receives Edith Creek, which also has many waterfalls along its course. ("Justo antes de entrar al valle recibe a Edith Creek, que también tiene muchas cascadas a lo largo de su curso.")*

Finalmente, existen en el corpus algunas oraciones etiquetadas como oraciones no especulativas, que parecen contener *weasel words*. Si examinamos el ejemplo 4.4, observamos que la primera oración no presenta información específica de los directores mencionados. Sin embargo, la oración inmediata provee la información necesaria para justificarla. De este ejemplo concluimos que la consideración del contexto por parte del anotador, puede llevar a posibles errores de anotación a nivel de oración.

Ejemplo 4.4 (falsos positivos - Contexto)

- *She has been singing since 1983 with many music directors. She worked with Telugu music directors like KOTI, MM Sreelekha, and M.M.Keeravani. ("Ha cantado con muchos directores musicales desde 1983. Trabajó con directores musicales Telugu como KOTI, MM Sreelekha, y M.M.Keeravani.")*

Examinando los falsos negativos, encontramos muchos ejemplos similares a los detectados exitosamente, pero que utilizan palabras distintas. En el ejemplo 4.5, observamos frases como *"regarded"*, *"greatest"* y *"much less"*, que no son detectadas como *weasel words*. Estas frases parecen ser candidatas adecuadas para extender la lista de wexes.

Ejemplo 4.5 (falsos negativos)

- *He is regarded as one of the greatest directors in cinema. ("Es considerado uno de los más grandes directores de cine.")*
- *Kamala beach to the north is much less developed. ("La playa Kalama al norte, está mucho menos desarrollada.")*

Confirmamos a través de este análisis, dos aspectos del problema que son frecuentemente mencionados en la literatura. Por un lado, observamos que una de las mayores dificultades en la detección de lenguaje especulativo es la correcta clasificación de los ejemplos positivos que no aparecen en el corpus de entrenamiento. Por otro lado, la falta de una definición formal y consensuada del fenómeno a detectar trae problemas de ambigüedad en algunos casos, ocasionando discrepancias en los criterios de anotación. Esto dificulta el desarrollo de algoritmos eficaces, dado que los corpus anotados son una pieza importante del mecanismo de aprendizaje.

Capítulo 5

Conclusiones y trabajo futuro

Este proyecto se concentró en el estudio del problema de detección de oraciones con contenido incierto mediante un enfoque activo de selección de ejemplos de entrenamiento para reducir los costos de anotación. Se construyeron y mejoraron dos clasificadores tradicionales alcanzando performance razonable con respecto al estado del arte. Sobre uno de ellos se construyó un clasificador que elige los ejemplos de entrenamiento basado en su seguridad de predicción. Luego de la evaluación y el proceso de análisis subsiguiente, es posible extraer algunas conclusiones con respecto al problema en sí por un lado, y con respecto a la metodología empleada por otro.

El problema de la detección de lenguaje especulativo, en particular la detección de oraciones con *weasel words*, es un problema que al día de hoy no se ha logrado resolver con un nivel de performance equivalente al alcanzado en otras tareas más establecidas como la detección de roles semánticos o el análisis sintáctico. Una de las mayores dificultades que identificamos es la detección de *weasel words* que no aparecen en el corpus de entrenamiento, situación que se ve agravada por la notoria variedad de expresiones utilizadas como *weasel words*. Esto influye negativamente sobre la performance en términos de exhaustividad de los clasificadores. Por otra parte, se detectaron algunos casos en los cuales algunas oraciones con *weasel words* aparecen en un contexto en el cual se justifica su contenido especulativo, dejando sin validez su clasificación como oración incierta. En este sentido, si pensamos en el objetivo último de la detección de *weasel words* como una forma de mejora de los procesos de extracción información, es posible que en algunos casos la clasificación a nivel de oración no sea suficiente, volviéndose necesaria la contemplación del contexto en que la oración ocurre.

Con respecto al *active learning*, los resultados que se obtuvieron en este trabajo indican que la aplicación exitosa es complicada en la práctica, y en general hay que cumplir con un conjunto de precondiciones que no son

siempre factibles. Entre ellas, es necesario que el conjunto inicial de ejemplos sea representativo del dominio entero, dado que en caso contrario se corre el riesgo de que las consultas subsiguientes se vean afectadas por el sesgo en el conjunto de partida. También es deseable que el oráculo no cometa errores, de otra forma se podrían estar descartando ejemplos equivocadamente. En nuestro caso el oráculo se simula través de un corpus anotado que contiene algunos discrepancias, mencionadas en el capítulo anterior. Finalmente, es importante también que se utilice un clasificador de base con buena performance. La factibilidad de este último punto depende de la tarea a resolver, ya que se puede querer aplicar *active learning* en problemas para los cuales aún no se conocen los mejores clasificadores.

No obstante estas dificultades, los experimentos llevados a cabo en este estudio no son concluyentes. Antes de poder argumentar a favor o en contra del *active learning* es preciso agotar las posibilidades de mejora que quedaron pendientes durante el desarrollo del proyecto. El ajuste del método de selección de consultas es un camino a explorar en este sentido. La aplicación de la selección por incertidumbre implementada en nuestro experimento tiende a favorecer la precisión, descuidando la exhaustividad. Sería interesante encontrar una forma de ajustar el cálculo de la incertidumbre de cada ejemplo, de forma de reducir la cantidad de ejemplos positivos descartados que luego se etiquetarán como negativos. Durante el proyecto se realizaron algunas pruebas con el objetivo de disminuir el sesgo de la selección que consistieron en incluir en cada lote seleccionado para consultar un subconjunto elegido de forma aleatoria. Los resultados que se obtuvieron fueron similares a los presentados en el capítulo anterior, por lo cual se concluye que es necesario evaluar con más profundidad el efecto de este tipo de decisiones en la selección de consultas. En la misma línea, se podrían investigar también otros métodos como la selección por desacuerdo.

Los resultados logrados parecen indicar que el aumento en la cantidad de ejemplos de entrenamiento impacta positivamente en la performance de los clasificadores. En este aspecto resulta interesante trabajar para obtener más ejemplos de entrenamiento. Por otro lado, sería favorable poder aprovechar la vasta cantidad de ejemplos parcialmente anotados que se pueden obtener directamente de Wikipedia. Se plantea como posible línea de trabajo futuro la elaboración de una herramienta de anotación que tome como entrada textos parcialmente anotados en el formato de Wikipedia y presente la información de forma amigable al anotador para clasificarlos. Adicionalmente, se podría elaborar una pequeña interfaz web que presente a los usuarios con estos ejemplos para clasificar. Si bien estos ejemplos no serían clasificados por expertos, se podría, por ejemplo, tomar para cada ejemplo la etiqueta más elegida por los usuarios, dejando al experto el trabajo de etiquetar aquellos

ejemplos más controversiales.

Si bien la performance alcanzada por el clasificador de base (NB-BOL) es razonable con respecto al estado del arte, aún es baja desde un punto de vista global. Resulta importante mejorar el clasificador de base para que alcance, al menos, resultados competitivos con respecto al estado del arte. La búsqueda de un conjunto de atributos que sea capaz de extraer características menos dependientes de las palabras en sí, podría resultar eficiente para mitigar la degradación de performance ocasionada por las *weasel words* que no aparecen en el corpus de entrenamiento. La inclusión de atributos referentes a la estructura sintáctica de las oraciones es una posibilidad que ha probado ser efectiva en trabajos similares. Estas mejoras también deberían impactar de forma positiva sobre los ejemplos que elige el método de selección de consultas, ya que se tendrían mejores atributos al momento del cálculo de la incertidumbre.

Finalmente, sería valioso contar con información de la performance de los clasificadores al ser entrenados y evaluados sobre otros dominios. El BioScope corpus es un candidato natural dado que tiene el mismo formato que el corpus utilizado en nuestros experimentos. Otro dominio interesante son las publicaciones en periódicos de noticias en dónde es frecuente encontrar *weasel words*. Esta segunda alternativa se comenzó a explorar mediante la elaboración de un script que extrae oraciones de las descripciones de noticias del New York Times, a partir de fuentes RSS. Una posible forma de trabajo para continuar esta idea involucra la anotación de un conjunto inicial de estos textos, para luego dejar que el algoritmo elija los nuevos ejemplos a etiquetar. Una vez completado el entrenamiento, mediante un umbral de performance o de cantidad de consultas, se podría proceder a evaluar el clasificador resultante.

Glosario

active learning Subárea de investigación del aprendizaje automático que pretende reducir los costos de entrenamiento mediante la elección inteligente de los ejemplos de entrenamiento. 1, 2, 5, 7, 11, 12, 16, 17, 19, 24, 32–35, 39, 42, 43, 49, 50

aprendizaje automático Área de la inteligencia artificial que se ocupa del desarrollo de algoritmos que mejoran su performance a través de la experiencia. 1, 4, 7, 8, 11, 17

aprendizaje supervisado Subárea del aprendizaje automático en donde se intenta aprender un clasificador a partir de ejemplos etiquetados. 2, 4, 16, 22, 24

Bag of Words Un forma de representar porciones de texto u oraciones en dónde no importa el orden en que aparecen las palabras. 21, 26

clasificador Una función que recibe una instancia representada por los valores de sus atributos y le asigna una categoría. 2, 5, 7, 10–12

corpus Una colección estructurada de textos sobre un mismo tema utilizados como entrada para algoritmos de aprendizaje automático. 1, 4, 5, 19–30, 32, 33, 36, 39–51

discriminativo Un tipo de clasificadores que modelan directamente la probabilidad $P(y|x)$ de una entrada x y una etiqueta y, o aprenden una función de asociación directa. 10

dump Respaldo o copia de la información de una base de datos a determinada fecha. 5

expresión regular Una expresión que se utiliza para representar un lenguaje o un subconjunto de tiras de un lenguaje. 29

falso negativo Ejemplo incorrectamente clasificado fuera de la categoría objetivo que un algoritmo está intentando detectar. 25, 44–48

falso positivo Ejemplo incorrectamente clasificado como perteneciente a la categoría objetivo que un algoritmo está intentando detectar. 25, 46, 47

generativo Un tipo de clasificadores que aprenden un modelo de la probabilidad conjunta $P(x, y)$, de la entrada x y la etiqueta y, utilizando el teorema de Bayes para calcular $P(y|x)$ y elegir la etiqueta y más probable. 10

hedging Denominación utilizada en inglés para el uso de lenguaje especulativo. 3

lema Forma canónica de una palabra. 28, 29, 32, 37

máxima verosimilitud Resultados de probabilidad en donde se asume que los ejemplos conocidos son los más probables. 9, 34

Naive Bayes Un tipo de clasificador lineal generativo que se basa en el teorema de Bayes. 7, 8, 10, 19, 30, 39, 43

oráculo Entidad capaz de proveer la respuesta correcta para una cierta tarea. 11–13, 16, 33, 50

SVM Máquinas de Vectores de Soporte. Un tipo de clasificador lineal discriminativo que busca maximizar la distancia entre los ejemplos de entrenamiento (representados como vectores) y un hiperplano de decisión. 7, 9, 11, 19, 24, 26, 27, 30, 32, 37, 39–42

tagger Algoritmo o software que asigna etiquetas a palabras. 37

token Unidad mínima de un lenguaje. 21, 26–28, 30, 32, 37

weasel word Un tipo de palabras que se utiliza en oraciones que contienen información especulativa. 2–5, 19–21, 23, 24, 28, 30, 42, 44–47, 49, 51

XML eXtensible Markup Language. Lenguaje de marco ampliable o extensible desarrollado por el World Wide Web Consortium (W3C). 23, 36, 37

Referencias

A. Beygelzimer, S. Dasgupta, and J. Langford. Importance weighted active learning. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 49–56, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1. doi: 10.1145/1553374.1553381. URL `http://dx.doi.org/10.1145/1553374.1553381`.

A. Beygelzimer, D. Hsu, J. Langford, and T. Zhang. Agnostic Active Learning Without Constraints. *CoRR*, abs/1006.2588, 2010.

D. Boswell. Introduction to support vector machines. 2002.

C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2: 27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

I. Dagan and S. Engelson. Commitee-based sampling for training probabilistic classifiers. In *Proceedings of the International Conference on Machine Learning(ICML)*, pages 150–157. Morgan Kauffman, 1995.

R. Farkas, V. Vincze, G. Móra, J. Csirik, and G. Szarvas. The conll-2010 shared task: learning to detect hedges and their scope in natural language text. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning — Shared Task*, CoNLL '10: Shared Task, pages 1–12, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. ISBN 978-1-932432-84-8.

V. Ganter and M. Strube. Finding hedges by chasing weasels: hedge detection using wikipedia tags and shallow linguistic features. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, ACLShort '09, pages 173–176, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1667583.1667636`.

M. Georgescul. A hedgehop over a max-margin framework using hedge cues. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 26–31, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W10-3004.

K. Hyland. *Hedging in scienctific research articles*. New series]. John Benjamins Publishing Company, 1998. ISBN 9789027250674.

D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000. ISBN 0130950696.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-778-1. URL http://dl.acm.org/citation.cfm?id=645530.655813.

G. Lakoff. Hedges: A study in meaning criteria and the logic of fuzzy concepts. *Journal of Philosophical Logic*, 2(4):458–508, Oct. 1973. doi: 10.1007/BF00262952. URL http://dx.doi.org/10.1007/BF00262952.

D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '94, pages 3–12, New York, NY, USA, 1994. Springer-Verlag New York, Inc. ISBN 0-387-19889-X. URL http://dl.acm.org/citation.cfm?id=188490.188495.

A. McCallum and K. Nigam. Employing EM in pool-based active learning for text classification. In *Proceedings of the International Conference on Machine Learning(ICML)*, pages 359–367. Morgan Kauffman, 1998.

T. M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997. ISBN 0070428077, 9780070428072.

R. Morante and C. Sporleder. Modality and Negation: An Introduction to the Special Issue. *Computational Linguistics*, pages 1–72, Feb. 2012. doi: 10.1162/COLI_a_00095. URL http://dx.doi.org/10.1162/COLI_a_00095.

R. Morante, V. Van Asch, and W. Daelemans. Memory-based resolution of in-sentence scopes of hedge cues. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning — Shared Task*, CoNLL '10: Shared Task, pages 40–47, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. ISBN 978-1-932432-84-8. URL http://dl.acm.org/citation.cfm?id=1870535.1870541.

B. Settles. Active Learning Literature Survey. Technical report, University of Wisconsin–Madison, 2009. URL http://www.cs.cmu.edu/~{}bsettles/pub/settles.activelearning.pdf.

B. Settles. From theories to queries: Active learning in practice. In *Active Learning and Experimental Design*, volume 15 of *JMLR Workshop and Conference Proceedings*, pages 1–18. Microtome Publishing, 2011.

B. Settles and M. Craven. An Analysis of Active Learning Strategies for Sequence Labeling Tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1069–1078. ACL Press, 2008. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.139.6640.

H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, COLT '92, pages 287–294, New York, NY, USA, 1992. ACM. ISBN 0-89791-497-X. doi: 10.1145/130385.130417. URL http://hebb.mit.edu/people/seung/papers/query.pdf.

G. Szarvas. Hedge classification in biomedical texts with a weakly supervised selection of keywords. In *ACL 08: HLT*, 2008.

B. Tang, X. Wang, X. Wang, B. Yuan, and S. Fan. A cascade method for detecting hedges and their scope in natural language text. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning — Shared Task*, CoNLL '10: Shared Task, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1870535.1870537.

S. Tong and E. Chang. Support vector machine active learning for image retrieval. In *Proceedings of the ninth ACM international conference on Multimedia*, MULTIMEDIA '01, pages 107–118, New York, NY, USA, 2001. ACM. ISBN 1-58113-394-4. doi: 10.1145/500141.500159. URL http://doi.acm.org/10.1145/500141.500159.

G. Tur, D. Hakkani-Tür, and R. E. Schapire. Combining active and semi-supervised learning for spoken language understanding. *Speech Communication*, 45(2):171 – 186, 2005. ISSN 0167-6393. doi: http://dx.doi.org/10.1016/j.specom.2004.08.002. URL `http://www.sciencedirect.com/science/article/pii/S0167639304000962`.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995. ISBN 0-387-94559-8.

X. Zhu, J. Lafferty, and Z. Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, pages 58–65, 2003.