Aprendizaje Activo para clasificación de preguntas sobre Datos Enlazados (Linked Data)

Teruel Milagro

Facultad de Matemática, Astronomía y Física

Universidad Nacional de Crdoba

Directores

Laura Alonso Alemany

Franco Luque

Índice general

Índice de figuras

Capítulo 1

Introducción

Los sistemas de respuesta a preguntas son un área naciente del procesamiento del lenguaje natural y particularmente del área de recuperación de información.

Gupta and Gupta [2012] destacan que existen dos formas principales de buscar la respuesta a una pregunta de un usuario. La primera de ellas consiste de encontrar similitudes semánticas o sintácticas entre la pregunta y documentos de texto que pueden contener evidencias para la respuesta. La segunda, que abordaremos durante este trabajo, traduce la pregunta a un lenguaje formal para luego realizar consultas a una base de datos.

La reciente posibilidad de manejo de grandes volúmenes de datos ha permitido la formación de grades bases de conocimiento públicas y disponibles online. Estas web semánticas u ontologías cambian ampliamente el paradigma utilizado hasta el momento, ya que estructuran los datos y permiten entonces extraer relaciones complejas entre sus entidades, como plantea Ou and Zhu [??].

Sin embargo, el primer paso para la resolución de una pregunta es la formalización de la misma a partir del texto ingresado por el usuario, independientemente del método de extracción de información empleado a continuación. Aún así, la mayoría de los sistemas se centran en la búsqueda de la respuesta más que en la correcta interpretación de la pregunta, y en general se limitan a textos cortos y preguntas puntuales.

Una aproximación simple a este problema es la de Quepy, un framework de traducción automática de preguntas en lenguaje natural a un lenguaje de consultas formalizado. El programador define una serie de plantillas para cada tipo de pregunta que el sistema pueda procesar y su correspondiente interpretación en la base de conocimiento elegida.

Aunque Quepy está diseñado para simplificar la tarea de construcción de dichas reglas, el trabajo necesario para lograr cobertura amplia es todavía prohibitivo por

varios motivos:

- Las plantillas deben ser desarrolladas por un experto de forma individual.
- El poder expresivo de las preguntas que soporte el sistema es lineal con respecto a la cantidad de plantillas generadas.
- Existe redundancia de información. Por ejemplo, para las preguntas "Who are the presidents of Argentina?" y "Who are the children of the presidents of Argentina?" se necesitan dos plantillas que contienen la misma información para resolver "presidents of X".
- Existen numerosas preguntas que son equivalentes y que no necesariamente se representan con la misma plantilla. Por ejemplo las preguntas "Where is Angelina Jolie from?" y "Where was Angelina Jolie born?" tienen escencialmente la misma semántica.
- Debido a las grandes variaciones del lenguaje natural, se requiere un anotador experto para lograr una cobertura completa de todas las reformulaciones para una misma semántica.

De todas las dificultades anteriores nos enfocaremos en las dos últimas ya que la consideramos prioritara y, al solucionarla, podemos ampliar la cobertura de los sistemas construidos sobre Quepy significativamente. Nuestra propuesta es aplicar un clasificador automático sobre las preguntas donde cada clase es una interpretación de Quepy. De esta forma, podemos ligar muchas más reformulaciones de la misma pregunta a su correspondiente semántica y lograr mayor versatilidad.

La originalidad de nuestra aplicación se base en utilizar como características las concordancias parciales con las plantillas de Quepy predefinidas por un programador. Consideramos que identifican claramente los aspectos relevantes que indican la correcta interpretación de la pregunta, y como tal son mejores representaciones.

Para evaluar nuestro sistema consideramos que comenzar con pocos patrones predefinidos nos ayudaría a percibir con más exactitud qué mejora podía generar el clasificador en Quepy. Por ello, y debido a que no existen grandes corpus etiquetados para reformulaciones de preguntas, planteamos que un enfoque de aprendizaje activo es lo más adecuado. El aprendizaje activo, como describe Settles [2009], permite entrenar el aprendedor con menor cantidad de instancias y es beneficioso cuando se cuenta con muchos ejemplos no etiquetadas pero donde la mayoría no son relevantes. En nuestro entorno en particular se da este fenómeno, debido a que en un corpus

no anotado estándar pocas de las preguntas caerán dentro de alguna de las clases semánticas de los patrones iniciales.

Un enfoque novedoso que combina todos los conceptos anteriores es el de Settles [2011] en Dualist. Esta herramienta optimiza el aprendizaje activo no solo preguntado al usuario sobre instancias sino también sobre características de las mismas que las asocian a una clase. Junto con este desarrollo también incluye una serie de investigaciones sobre el rendimiento de tareas de clasificación con usuarios reales y simulados. Es por ello que tomamos como base este trabajo y lo adaptamos con una nueva implementación a nuestro problema.

Capítulo 2

Definición formal del problema

Tanto el problema que planteamos abordar como la solución propuesta son complejos de definir, ya que incluye numerosos conceptos del procesamiento de lenguaje natural. En la primera parte de esta sección definiremos un marco teórico para cada aspecto no central del problema. A partir de esta base, en la segunda parte daremos una definición propiamente dicha, seguida por una formalización de la solución.

2.1. Datos Enlazados y Sistemas de Respuesta

La cantidad de infomación disponible en internet es abrumadora, y sin embargo, aún no puede utilizarse en conjunto para extracción de infomación satisfactoriamente. Berners-Lee [2014] explican que este fenómeno se debe a que fragmentos de información que se refieren al mismo objeto o suceso no están relacionados entre sí

Christian Bizer and Berners-Lee [2009] definen los datos enlazados como infomación que cumple las siguientes características:

1. Puede ser leída automáticamente por una computadora.
2. Su significado está explícitamente definido.
3. Está conectada a fuentes de datos externas.
4. Puede ser conectada desde fuentes de datos externas a su vez.

Sin embargo, no existe un conceso o una definición formal sobre el tema. Berners-Lee [2014] describe en su artículo un protocolo orientativo para publicar datos enlazados en la web de tal forma que pudiera formase una base de conocimiento global. Con el tiempo estas reglas se han tomado como un estándar para la construcción de ontologías, y en la actualidad existen espacios de información que contienen millardos de aserciones del mundo real.

Los datos enlazados se representan comunmente como una colección de tripletas siguiendo un lenguaje de descripción como RDF, tal como lo describe Brickley and Guha [2014]. Cada tripleta se compone de un sujeto, un predicado y un objeto, donde el predicado representa una relación entre el sujeto y el objeto. De esta forma se puede representar cualquier tipo de asociación entre entidades sin importar su complejidad, contruyéndolo a partir de relaciones parciales. El resultado es información organizada en forma de grafo donde cada nodo es una entidad y cada arista es una relación entre dichas entidades.

Las web semánticas u ontologías más populares en el momento son FreeBase [1] y DBPedia [2], aunque existen numerosos proyectos con dominios más acotados como WordNet [3]. Estas plataformas son abiertas con interfaces fáciles de utilizar que permiten agregar nuevos datos, y como resultado se observa un rápido crecimiento en la cantidad de información disponible.

Estos sitios cuentan con puertos de accesos donde los usuarios pueden enviar consultas utilizando algún lenguaje formal. Aunque este servicio es accesible para cualquier persona, se requiere cierto nivel de conocimiento técnico para generar dichas consultas. Para dar acceso real a las masas a esta gran cantidad de información de requieren interfaces capaces de extraer datos a partir de consultas en lenguaje natural, es decir, sistemas de respuestas a preguntas.

Paralelamente, los sistemas de respuesta a preguntas pueden obtener grandes beneficios de una ontología. En lugar de buscar documentos o pasajes que puedan contener una respuesta, los datos enlazados pueden brindar información exacta. Además de ello, resulta más fácil procesar preguntas donde es muy poco probable encontrar la respuesta en un solo documento, por ejemplo, "¿Qué famosas actrices nacieron en el mismo país que Naima Akef?". Desde los años 70 este tipo de software ha utilizado bancos de conocimiento estructurada que inicialmente eran bases de datos locales. Sin embargo, los resultados obtenidos no se destacaron particularmente. Con el desarrollo de las nuevas web semánticas la atención ha vuelto nuevamente hacia los datos relacionados.

Extraer información de una ontología no es difícil, sin embargo, como describe Unger et al. [2014], indentificar el sector de datos relevante a una consulta en lenguaje natural es un gran desafío. Se requiere para esto traducir el texto ingresado por el usuario en una consulta formal que pueda ser procesada por un motor de búsqueda

[1] www.freebase.com
[2] www.dbpedia.org
[3] www.wordnet.princeton.edu

tradicional sobre datos enlazados. Una vez que se ha obtenido la información de la base, otra etapa de procesamiento convierte estos datos del formato legible por una computadora a un formato legible por el usuario. A continuación ilustramos con un ejemplo estas etapas utilizando una consulta en lenguaje MQL sobre la estructura de FreeBase.

Ejemplo 1.

1. Obtención de la pregunta.

```
What is the capital city of Argentina?
```

2. Generación de la consulta MQL.

```
{
    "type":"/location/country",
    "id":"/en/argentina",
    "capital":null
}
```

3. Obtención de la información.

```
{
    "result": {
        "capital": "Buenos Aires",
        "type": "/location/country",
        "id": "/en/argentina"
    }
}
```

4. Generación de la respuesta en leguaje natural.

```
The capital city of Argentina is Buenos Aires.
```

El primer desafío es identificar la entidad a la que se hace referencia en la pregunta, en nuestro caso, "Argentina". Esta tarea se complicaría con nombre más complejos como "People's Republic of China". Las complicaciones de este etilo está ligadas a los sistemas externos de parseo y asignación de etiquetas morfosintácticas. Sin un buen procesamiento del lenguaje natural poco puede contruirse.

Adicionalmente, las consultas contienen no sólo información brindada por la pregunta del usuario, sino también datos asociados a la estructura de la base. Si en lugar de "/location/country" hubieramos utilizado "/location/location" la consulta hubiera devuelto un error, a pesar de que Argentina es también de tipo "/location/location".

Unger et al. [2014] menciona también otros problemas que frecuentemente enfrentan este tipo de sistemas.

En las consultas utilizando MQL se detalla la estuctura de la información y se completan los datos necesarios para identificar el objeto en la base de datos. Para obtener información sobre la entidad se nombran sus atributos, pero se les da un valor de *null*. El motor de búsqueda identifica estos campos y completa la información faltante. Este lenguaje es muy intuitivo y fue diseñado para ser accesible, pero no todos los lenguajes de consulta son tan simples como MQL.

Ejemplo 2. Consulta en SPARQL para la pregunta "How many episodes does Seinfeld have?"

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpprop: <http://dbpedia.org/property/>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>

SELECT DISTINCT ?x1 WHERE {
  ?x0   rdf:type                      dbpedia-owl:TelevisionShow .
  ?x0   dbpprop:showName              "Seinfeld"@en.
  ?x0   dbpedia-owl:numberOfEpisodes  ?x1.
}
```

La cantidad de información necesaria para construir esta consulta es mucho mayor mientras que su estructura no es simple de comprender. Sin embargo, pone en relevancia el uso de tripletas para representar la relación entre distintos nodos. En particular, la variable $?x1$ representa el resultado, mientras que la variable $?x0$ representa a la entidad de nombre "Seinfield" y tipo "TelevisionShow".

Cómo cierro esta sección? Quiero dar una buena introducción a las consultas para después explicar bien cómo funciona Quepy.

2.2. Quepy

Como se mencionó anteriormente, Quepy es un marco de trabajo para crear aplicaciones de respuesta a preguntas. Su objetivo principal es brindar una herramienta fácilmente adaptable a distintos dominios y distintos lenguajes de consultas. Los lenguajes soportados hasta el momento son MQL y SPARQL; ambos permiten consultas posteriores a FreeBase y DBPedia. Haremos un breve resumen a continuación sobre la arquitectura general de Quepy y sus principales características.

Una aplicación creada en Quepy tiene tres secciones principales:

Settings La configuración de Quepy incluye las herramientas de análisis sintáctico a utilizar, la URL del servidor para enviar las consultas, etc.

Templates Contiene las plantillas definidas por el creador de la aplicación. Cada plantilla es una expresión regular que combina distintos tipos de caracteríticas como etiquetas POS y lemmas, lo que permite al sistema identificar la semántica de la pregunta únicamente en base a su sintáxis. Junto con la expresión regular, cada plantilla tiene una función de interpretación que toma las secciones de la pregunta que considera relevantes y las utiliza para construir una representación interna de la pregunta llamada Expresión.

DSL Son las siglas correspondientes a Lenguaje de Dominio Específico en inglés. En esta selección se detalla cómo las Expresiones de Quepy se traducen a las partes integrantes de una consulta formal.

A grandes rasgos, Quepy utiliza dos etapas que traducen una pregunta a una Expresión y luego utilizan la Expresión para formar consultas. Esto es así ya que permite soportar diversos lenguajes de consultas. Estas representaciones internas son generales y pueden generar cualquier consulta. Es el programador quien se encarga de especificar las reglas de construcción de las expresiones y las de traducción a lenguaje formal, por ejemplo SPARQL.

2.2.1. Construcción de las consultas

Para entender mejor cómo funciona Quepy internamente veamos en ejemplo en particular, extraído de la documentación oficial [4]. Este ejemplo corresponde a una aplicación realizada para generar consultas SPARQL para ser enviadas a un motor de la DBPedia. Analicemos primero cómo se definen los elementos del DSL para luego seguir con las plantilla propiamente dichas.

Ejemplo 3. Definición de un elemento del DSL.

```
from quepy.dsl import FixedRelation

class IsDefinedIn(FixedRelation):
    relation = "rdfs:comment"
    reverse = True
```

[4] http://quepy.readthedocs.org/en/latest/tutorial.html

La clase $IsDefinedIn$ es una Expresión que representa una relación entre dos objetos, como vimos anteriormente en RDF. Dependiendo del lenguaje de consulta tendrá distinas traducciones, y en particular para SPARQL es equivalente a:

```
?target rdfs:comment ?definition
```

donde ?$target$ y ?$definition$ son parámetros que tomará la Expresión al instanciarse.

Las expresiones pueden construirse progresivamente a partir de otras expresiones como veremos a continuación.

Ejemplo 4. Plantilla para las preguntas de tipo "What is ... ?".

```
from refo import Group, Question
from quepy.dsl import HasKeyword
from quepy.parsing import Lemma, Pos, QuestionTemplate

from dsl import IsDefinedIn

class WhatIs(QuestionTemplate):

    aux = Question(Pos("DT")) + Group(Pos("NN"), "target")
    regex = Lemma("what") + Lemma("be") + aux + Question(Pos("."))

    def interpret(self, match):
        thing = match.target.tokens
        target = HasKeyword(thing)
        definition = IsDefinedIn(target)
        return definition
```

Observemos que la clase tiene un attributo llamado *regex* que corresponde a la expresión regular que define la plantilla. Profundizaremos en la estructura de estas expresiones regulares más adelante, pero ahora notemos que uno de los elementos tiene una etiqueta *target*. Si la pregunta ingresada por el usuario concuerda con esta expresión regular, entonces los elementos que concuerden con las sub expresiones etiquetadas serán pasados al método *interpret* de la clase. En este caso, el segmento de oración que corresponda a $Group(Pos("NN"))$ (un conjunto de sustantivos) será un atributo del parámetro *match* recibido por *interpret*.

El método *interpret* construye una Expresión de tipo $HasKeyword$ a partir de *target* y luego la utiliza para contruir otra Expresión de tipo $IsDefinedIn$. El resultado final de la Expresión traducida a SPARQL para la pregunta "What is a car?" será:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX quepy: <http://www.machinalis.com/quepy#>

SELECT DISTINCT ?x1 WHERE {
  ?x0 quepy:Keyword "car".
  ?x0 rdfs:comment ?x1.
}
```

2.2.2. Plantillas y sus expresiones regulares

Describiremos a continuación más en detalle la estructura de las plantillas que permiten crear una Expresión a partir de una pregunta. Cada una de las plantillas está construida en base a la librería REfO[5], que define expresiones regulares entre objetos complejos de Python, no solamente cadenas de caracteres.

Ejemplo 5. Expresión regular del ejemplo 4[6].

```
regex = Lemma("what") + Lemma("be") + Question(Pos("DT"))
        + Group(Pos("NN"), "target") + Question(Pos("."))
```

Para analizar si un frase concuerda o no con una expresión regular, Quepy transformará la oración con el analizador sintáctico indicado para obtener el lemma y las etiqueta POS de cada una de sus palabras. Luego, utilizará esa información para compararla con la expresión regular. Entonces, nuestro ejemplo concordará con una frase cuya primera palabra tenga lemma "what", su segunda palabra tenga lemma "be", su tercera palabra (opcionalmente) tenga etiqueta POS "DT", etc.

Dada una pregunta, Quepy intentará encontrar una concordancia con cada una de estas expresioner regulares existentes. Si la encuentra, entonces utilizará el método *interpret* que explicamos en la sección anterior para construir una Expresión y luego una consulta.

Definir patrones de reconocimiento de esta manera permite representar tanto información semántica como sintáctica y por lo tanto tienen mayor poder expresivo que cualquiera de ellas por separado. Veremos más adelante que elegimos estos patrones como una forma de representación de las preguntas para clasificar.

[5] https://github.com/machinalis/refo

[6] Reemplazamos la variable aux por su contenido para mayor claridad, lo cual no afecta el significado de la expresión regular.

2.3. Formalización del problema

A pesar de las numerosas ventajas de Quepy, también existen desventajas. La más importante de ellas es que, al utilizar expresiones regulares, los patrones no tienen flexibilidad y dependen fuertemente del analizador sintáctico y POS tagger que utilicen.

En particular, si tomamos el ejemplo de la sección anterior, no se podrían reconocer preguntas del estilo "Definition of a car" o "How would you define what a car is?". La respuesta a estas preguntas se obtiene con la misma consulta generada que acabamos de analizar, por lo cual son escencialmente equivalentes. Diremos entonces que estas preguntas comparten la misma semántica, y que son reformulaciones una de la otra.

Para agregar un nuevo tipo de pregunta al sistema se deben definir sus patrones y sus traducción a una consulta. Gracias a la gran cantidad de formas distintas en las que se puede expresar una pregunta es imposible construir todas las expresiones regulares necesarias, y los sistemas de Quepy están fuertemente limitados por esta característica. Si los patrones fueran más generales o pudieran inferirse de alguna forma, entonces ampliar los tipos soportados consistiría sólo en el segundo paso.

Unger et al. [2014] clasifica los sistema de respuesta a preguntas sobre datos enalzados (QALD por sus siglas en inglés) según sus estrategias de resolución de la pregunta. Entre ellos se encuentra la clase a la cual pertenece Quepy, llamada por los autores "Template-Based approaches" o Aproximaciones Basadas en Patrones. Claramente, la falta de cobertura sobre el universo posible de preguntas en una dolencia de cualquier sistema que utilice patrones estáticos para clasificar las preguntas en una determinada representación.

Lo que nos proponemos entonces lograr con este trabajo es ampliar la cobertura de un sistema QALD basado en concordancia con patrones para reconocer preguntas semánticamente equivalentes a una de sus clases ya definidas en él. El sistema QALD que tomamos como base es Quepy, en particular una aplicación realizada como demostración del producto[7]. A partir de este punto, utilizaremos la palabra Quepy para referirnos tanto al marco de trabajo como a las aplicaciones construidas por él, y en particular a la que estaremos usando.

[7]Puede utilizarse online ingresando a http://quepy.machinalis.com/

2.4. Solución propuesta

Como la generación de nuevas plantillas manualmente no es viable, entonces proponemos una solución automática: agregar al sistema un clasificador que identifique (si existiera) el patrón que corresponde a la pregunta. Es tarea del clasificador asociar reformulaciones que tengan la misma semántica a solo patrón. Una vez obtenida la clase semántica e identificado el objeto de la pregunta, Quepy u otro sistema puede construir la consulta directamente. Dejaremos como trabajo futuro el reconocimiento de la entidad base y nos centraremos en la clasificación de las preguntas.

Este enfoque de encontrar reformulaciones de una misma pregunta está enmarcado dentro del reconocimiento de implicaciones textuales y ha sido utilizado previamente para sistema de respuesta a preguntas del usuario. Ou and Zhu [??] utilizan esta técnica tomando como base preguntas modelo construidas automáticamente desde la ontología, y se centran también en la composición de patrones simples para formar otros más complejos. Sin embargo, se limitan a un dominio muy restringido que permite formar texto en lenguaje natural desde las relaciones formales entre las entidades, lo cual sería dificultoso en ontologías complejas como FreeBase. Wang and Li [??] explican otros posibles usos de identificar estas relaciones entre las preguntas para sugerencia de preguntas relacionadas o útiles para el usuario. El trabajo de Kosseim and Yousefi [2008], por otra parte, utiliza la reformulación para obtener patrones semánticamente equivalente, pero utiliza durante el entrenamiento del clasificador la respuesta de la pregunta.

Nuestro trabajo será construir y entrenar un clasificador capaz de recibir una pregunta y decidir a qué clase semántica pertenece, siguiendo la definición de Sebastiani [2002]:

Definición 1. La clasificación de una instancia es una función $\Psi : \mathcal{X} \times \mathcal{C} \to \{0, 1\}$ que asigna valores booleanos donde $\mathcal{X}$ es el dominio de las instancias y $\mathcal{C}$ es el conjunto de clases posibles.

Asignaremos el valor 1 o $Verdadero$ a los pares $\langle x_i, c_j \rangle$ si la clase c_j corresponde a la instancia x_i, y 0 o $Falso$ en el caso contrario. Como en nuestro caso la clase asociada a cada instancia es única, podemos utilizar la siguiente definición:

Definición 2. Un clasificador monoclase es una función $\Phi : \mathcal{X} \to \mathcal{C}$ tal que:

$$\Phi(x_i) = c_j \Leftrightarrow \Psi(x_i, c_j) = 1$$

$\mathcal{C}$ para esta clasificación es el conjunto de clases semánticas de Quepy, es decir, cada una de las plantillas o patrones. El ejemplo que describimos en la sección anterior corresponde a la clase "Whatis". Todas las preguntas que puedan responderse a través de la consulta generada por esta plantilla serán clasificadas dentro de esta clase. La cantidad total de clases es 29, las agregamos en un APENDICE?

Aunque la tarea a realizar no parece compleja y ha sido ampliamente estudiada, nos encontramos con numerosos obstáculos que impiden utilizar algún método estándar de clasificación de texto. A continuación discutiremos dos de estos incovenientes y las decisiones que tomamos para resolverlos.

2.4.1. De la Clasificación Supervisada al Aprendizaje Activo

En primer lugar, no contamos con un corpus anotado que permita utilizar clasificación supervisada común. Desarrollamos entonces un pequeño corpus a partir de los ejemplos incuidos en Quepy. Por cada clase agregamos también algunos casos de reformulaciones no reconocidos por la aplicación y también las etiquetamos. El resultados final fueron 106 preguntas, un número más que modesto y que difícilmente cubre el universo de posibles reformulaciones de todas las clases.

Sin embargo, existen numerosos corpus de preguntas utilizados para otras tareas de clasificación que no están etiquetados. Por lo tanto, decidimos utilizar un enfoque semiautomático que comience con un conjunto de semillas y que utilice las preguntas no etiquetadas paulatinamente para aprender la clasificación. Esto nos permitirá compensar la falta de cobertura sobre el dominio.

La fuente más importante de preguntas para la construcción del corpus no anotado fueron las preguntas de entrenamiento y evaluación de las tareas del TREC[8] desde el año 2008. Por lo tanto, consideramos que nuestro conjunto representativo de las posibles preguntas que un usuario podría esperar que un sistema responda. Sin embargo sólo una porción muy pequeña del ellas se corresponde con alguna de las clases de Quepy. Por lo tanto, entrenar un casificador con tan alta cantidad de ruido sería una tarea muy difícil.

Tengamos en cuenta también que los límites de una clase semántica no siempre están claros y algunas veces dependen fuertemente de la estructura de los datos en la ontología.

Ejemplo 6.

1. "What is the tallest mountain?"

[8] http://trec.nist.gov/data/qamain.html

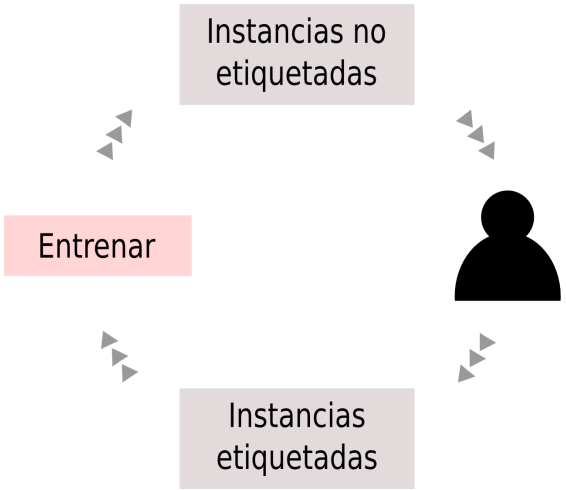
2. “What is the Everest mountain?”

Estas preguntas son muy similares, y sin embargo sólo la segunda pertenece a la case “whatis”: para responder la primera pregunta debe obtenerse la altura de todas las montañas de la base de datos y seleccionar la mayor.

Por este motivo decidimos utilizar una plataforma de aprendizaje activo donde un oráculo humano ayudará al sistema a delimitar estas sutilezas semánticas. Hospedales et al. [2011] y Ertekin et al. [2007] describe clasificadores adaptados a través del aprendizaje activo para encontrar y clasificar ejemplos raros de clases minoritarias. Además de ello, el aprendizaje activo es una estrategia que obtiene buenos resultados para problemas con una gran cantidad de clases, de acuerdo con Jain and Kapoor [2009].

Settles [2009] explica que el aprendizaje activo es un paradigma donde el aprendedor selecciona preguntas para que un humano u oráculo las etiquete. Si el aprendedor elige las instancias de las cuales puede aprender más información, entonces se minimiza la cantidad de instancias etiquetadas necesarias para lograr el mismo desempeño. La mayor motivación para este tipo de estrategias es la dificultad de conseguir dato etiquetados al mismo tiempo que se disponen de grandes cantidad de ejemplos no etiquetados, tal y como es nuestro caso. Utilizaremos aprendizaje activo para conseguir un corpus etiquetado de entrenamiento con el menor esfuerzo posible.

Figura 2.1: Arquitectura de un sistema de aprendizaje activo



2.4.2. Aprendizaje activo sobre características

Durante las primeras etapas de desarrollo y especificación del problema debimos definir la representación de las instancias ante el clasificador. Sin embargo, al no existir trabajos previos con la misma proximación al problema no tenemos un punto de referencia para tomar como ejemplo. Por esto, decidimos incluir características tentativamente y realizar experimentos con aprendizaje activos sobre características e instancias.

En un enfoque como este se pedirá al usuario que etiquete las características seleccionadas con una clase si la presencia de la característica en una instancia es indicativa de la clase. Druck et al. [2009] han realizado experimentos sobre clasificación de texto en donde se demuestra que el aprendizaje activo sobre características puede dar mejores resultados incluso que el aprendijase sobre instancias. Durante este trabajo ayudará a identificar las características que mejor describen las instancias para la clasificación descripta.

Las etiquetas obtenidas se utilizarán también para entrenar el clasificador reforzando la probabilidad de una característica dada una clase, como veremos en detalle en la implementación.

2.4.3. Dualist

Dualist es un sistema muy similar al que estamos planteando desarrollado por Settles [2011] que combina el aprendizaje sobre instancias y sobre características. Los resultados presentados son para diversas tareas como el análisis de sentimientos o la clasificación de documentos.

La interfaz gráfica de una instancia tiene dos secciones principales. A la izquierda se muestra una lista de instancias con las clases para que el usuario pueda etiquetarlas sólo con un click. A la derecha, por cada clase hay una lista de objetos seleccionables representando las características (en este caso palabras) que están potencialmente asociadas a la clase. Settles [2011] sostienen que presentar al usuario toda la información en conjunto hará que éste etiquete una mayor cantidad antes de esperar a que el clasificador de reentrene o le presente nuevas opciones.

Nuestra implementación seguirá los mismos lineamientos principales y pondremos en práctica las técnicas de selección de ejemplos con las de entrenamiento del clasificador para obtener resultados sobre nuestra configuración. Decidimos tomarlo como modelo ya que se centra en la interacción con el usuario y en la capacidad del software de ser utilizado en tiempo real. Nosotros deseamos lograr un sistema que sea viable

Figura 2.2: Captura de pantalla de la interfáz gráfica de Dualist.

de integrar con Quepy y complementar aplicaciones reales, en lugar de ser utilizado sólo para demostrar los resultados de este trabajo.

Capítulo 3

Representación de las preguntas

Una importante parte de cualquier problema que aborde el lenguaje natural es la representación del mismo. Las características relevantes son propias de cada problema, si bien existen numerosos ejemplos bibliográficos a tomar como modelo. El siguiente paso es identificar qué características de la pregunta son indicativas de su clase semántica, lo cual no tiene una respuesta intuitiva.

Nuestra primera aproximación fue utilizar criterios estándares en la catergorización de texto como los lemmas de las palabras y sus etiquetas POS. Sebastiani [2002] describe que a pesar de su simpleza son representaciones muy poderosas y que otras más complejas no necesariamente llevarán a un mejor desempeño del clasificador.

3.1. Subpatrones

Como presentamos en el ejemplo 6 algunas preguntas tienen tanto lemmas como etiquetas POS muy similares y sin embargo pertencen a clases distintas. La forma en la que Quepy distingue estos casos es utilizando la estructura de la frase, representada en sus patrones. Por eso, decidimos utilizar además como característica los patrones que concuerdan con la pregunta.

Esta representación por sí sola tampoco mejora nuestra situación inicial, ya que sólo reconoce las preguntas que corresponden a un patrón. La solución que encontramos para este problema fue dividir cada patrón en todos sus posibles subpatrones y asignar a cada instancias la concordancia con los subpatrones.

Ejemplo 7. Subpatrones del ejemplo 5.

1. `Lemma("what") + Lemma("be") + Question(Pos("DT"))`
   `+ Group(Pos("NN"), "target") + Question(Pos("."))`

2. `Lemma("what") + Lemma("be") + Question(Pos("DT")) + Group(Pos("NN"), "target")`

3. `Lemma("what") + Lemma("be") + Question(Pos("DT"))`

4. `Lemma("what") + Lemma("be")`

5. `Lemma("what")`

6. `Lemma("be")`

7. `Lemma("be") + Question(Pos("DT")) + Group(Pos("NN"), "target") + Question(Pos("."))`

8. ...

3.2. Nombres y tipos de entidades

Existe un tipo más de característica que determina fuertemente la semántica de la pregunta. La forma de acceder a una propiedad de una entidad dentro de una base de datos está íntimamente ligada a la estructura que le dió el usuario al ingresar los datos. Por lo tanto, la misma propiedad como "fecha de creación" puede tener dos nombres distintos para tipos de entidades. Por lo tanto, dependiendo de un tipo o de otro la consulta que debe generarse es dinstinta, y por eso son necesarias dos clases semánticas distintas.

Ejemplo 8. Ejemplos con semántica diferenciada por el tipo de la entidad.

1. "Who are the actors of Titanic?"

2. "Who are the actors of Friends?"

Para obtener los actores que trabajaron en una película, como en el caso de la primera pregunta, debe utilizarse la relación "/film/film/starring", mientras que en el caso de una serie televisiva se utiliza "/tv/tv_program/regular_cast".

El único indicio de la clase semántica en estos casos es la entidad referenciada. Por ello, la incluímos como una característica más. Adicionalmente, agregaremos los tipos de dicha entidad en la base de conocimiento, particularmente para esta aplicación FreeBase. Cabe destacar que no todas las preguntas tienen una entidad, y en el caso

de que sí tenga no siempre podemos reconocerla. Esto depende del sistema externo de reconocimiento de nombres de entidades o NER por sus siglas en inglés.

En resumen, las características propuestas para el sistema son:

- Etiquetas POS.
- Lemmas.
- Concordancias a patrones parciales.
- Nombre de la entidad referenciada.
- Tipos de la entidad referenciada.
- Bigramas y trigramas.
- Bigramas mezclando Lemmas y POS.

Capítulo 4

Implementación

4.1. Arquitectura del sistema

Si bien el objetivo principal de este trabajo es incrementar la cobertura de Quepy, deseamos también que el sistema esté compartimentado de tal forma que sus componentes puedan utilizarse individualmente para otras aplicaciones.

La arquitectura de nuestro sistema integra tres grandes bloques que interactúan a través de interfaces:

FeatMultinomialNB Es el clasificador del sistema. Hereda del clasificador multinomial bayesiano *MultinomialNB* de la librería scikit-learn desarrollada por Pedregosa et al. [2011] y está adaptado para soportar el entrenamiento utilizando tanto instancias como características. Agregamos algunos métodos auxiliares más a la clase que simplifican algunos cálculos para el aprendizaje activo.

ActivePipeline Es una clase que abstrae los ciclos de aprendizaje activo. Recordemos que constan de dos pasos principales donde se seleccionan las instancias (o características) y luego se procesan las etiqueta devueltas por el usuario. Para llevar a cabo estas tareas el pipe necesita tener acceso a los corpus etiquetados y no etiquetados y al clasificador, lo cual lo convierte en el módulo central del proyecto.

Preprocess Es un módulo que convierte las preguntas de entrenamiento etiquetadas a su representación matricial utilizando las características descriptas en el capítulo anterior. La representación de las preguntas está completamente ligada a los patrones de la aplicación de Quepy. Por ello diseñamos el preproceso como una extensión opcional de Quepy que puede utilizar cualquier clasificador. En otras palabras, no es necesario que se integre con aprendizaje activo.

4.1.1. ActivePipeline: Un marco de trabajo de aprendizaje activo sobre características e instancias

ActivePipeline es una clase creada para simplificar la tarea del aprendizaje activo. Entre las actividades que realiza se encuentra la lectura de los corpus, la selección de instancias y características para presentar al usuario, la gestión de los datos ingresados y el reentrenamiento del clasificador. También agregamos funciones que no eran necesarias pero facilitan el entorno de experimentación como sessiones y métricas de evaluación parcial.

Hasta el momento hemos realizado la prueba de concepto que demuestra que una aproximación de este estilo ayudaría a resolver el problema de la falta de cobertura de Quepy. Esta arquitectura está en desarrollo constante y no ha sido pulida para interactuar con un sistema real. Describiremos a continuación los puntos más centrales de la implementación ya que los detalles son altamente propensos a ser modificados en el futuro.

Para crear una instancia de ActivePipeline se requiere un diccionario con los siguientes parámetros:

Clasificador Es una instancia de un clasificador. Para mantener generalidad, requerimos que tenga la interfaz estándar de sklearn. Los métodos que utilizamos son *fit*, *predict_proba*, *predict_log_proba* y *score*.

Corpus Se deben definir los archivos desde donde recuperar al menos tres corpus: el de entrenamiento, el de evaluación y el no etiquetado. Los corpus ya deben estar procesados antes de ser leídos por el ActivePipeline, y es recomendable que sean instancias de una clase Corpus también definida en el sistema.

Al permitir elegir tanto características como el corpus y el clasificador, la clase ActivePipeline puede ser utilizada dentro de cualquier ámbito incluso no relacionado al procesamiento del lenguaje natural.

Tanto para el aprendizaje sobre características y sobre instancias el ActivePipeline cuenta con una función automática. El usuario debe definir sólo las funciones de interacción con el usuario, es decir, cómo mostrar la información, y pasarlas como parámetros al ActivePipeline.

4.1.2. FeatMultinomialNB: Un clasificador entrenado con características

Como ya mencionamos anteriormente FeatMultinomialNB es un clasificador bayesiano ingenuo. Profundizaremos ahora en el fundamente teórico detrás de él y en las modificaciones que realizamos para soportar el entrenamiento con características.

Un clasificador bayesiano, explica Abney [2007], es un ejemplo de un modelo generativo. Estos modelos usualmente utilizan la distribución anterior de las clases $P(c)$ y la distribución de las instancias específica para la clase o verosimilitud $P(x|y)$. Su nombre se debe a que un modelo es una hipótesis acerca de cómo cada instancia puede ser generada por una clase. Se utiliza esta probabilidad generativa para la clasificación a través del teorema de Bayes:

$$P(x|c) = \frac{P(c)P(x|y)}{P(x)}$$

El *MultinomialNB* descripto por Manning et al. [2008] se basa en la asunción ingenua de que el valor de cada una de las características de una instancia es independiente de las demás. Si bien sabemos que esto no es así y que cada característica sí se relaciona con las restantes, este clasficador ampliamente utilizado para la categorización de texto Settles [2011], McCallum and Nigam [1998] y Frank and Bouckaert [2006]. Su principal ventaja es la simplicidad, lo que en nuestro caso ayudó en la tarea de la modificación.

En gran parte de la bibliografía se asume que las características de una instancia serán sólo las palabras presentes en el documento o instancia, lo cual no se ajusta a la representación elegida para este problema. Por lo tanto, hemos cambiado levemente la notación que utilizan otros autores reemplazando palabras como *words* o *terms* por *características*. También hemos reemplazado *documents* por *instancias*.

Bajo este modelo, la probabilidad de que una instancia x_i sea de clase c_j es computada utilizando la fórmula:

$$P(c_j|x_i) \propto P(c_j) \prod_{1 \leq k \leq n_i} P(f_k|c_j) \tag{4.1}$$

donde n_i es la cantidad de características que aparecen en la instancia x_i y $P(f_k|c_j)$ es la probabilidad de que la característica f_k esté presente en una instancia de clase c_j. Manning et al. [2008] explican que la intuición detrás de estos conceptos es que $P(f_k|c_j)$ indica de qué tan buen indicador es f_k para la clase c_j, mientras que $P(c_j)$ pesa estos valores por la probabilidad de la clase. El producto de las probabilidades

y pesos es una medida de cuánta evidencia existe de que la instancia pertenezca a la clase.

La gran cantidad de multiplicaciones sobre probabilidades menores que uno puede llevar fácilmente a un desbordamiento aritmético debido la imposibilidad de representar números tan pequeños en una computadora. Utilizaremos logaritmos para manejar esta situación, manteniendo la proporción de los operandos ya que el logaritmo es una función monótona creciente y basándonos fuertemente en la propiedad que asegura $\log(ab) = \log(a) + \log(b)$. La ecuación 4.1 es equivalente a:

$$\log P(c_j|x_i) \propto \log P(c_j) + \sum_{1 \leq k \leq n_i} \log P(f_k|c_j) \tag{4.2}$$

La tarea de clasificación se reduce entonces a encontrar la clase c_j que maximice la ecuación 4.1. Volviendo a la definición 2, podemos reescribirla como:

Definición 3.

$$\Phi(x_i) = \underset{c_j \in \mathcal{C}}{\operatorname{argmax}}\ \log P(c_j|x_i) = \underset{c_j \in \mathcal{C}}{\operatorname{argmax}}\ \log P(c_j) + \sum_{1 \leq k \leq n_i} \log P(f_k|c_j) \tag{4.3}$$

Nuestro modelos depende entonces de dos parámetros: $P(c_j)$ y $P(f_k|c_j)$. Sin embargo no conocemos las distribuciones reales de estas variables, y por lo tanto las estimaremos empíricamente a partir de los datos observados. Llamaremos $\hat{P}(c_j)$ y $\hat{P}(f_k|c_j)$ a las estimaciones respectivamente, que se calculan utilizando estimadores de máxima verosimilitud:

$$\hat{P}(c_j) = \frac{\sum_{x \in \mathcal{L}} \hat{\Psi}(x, c_j)}{|\mathcal{L}|} \tag{4.4}$$

$$\hat{P}(f_k|c_j) = \frac{\sum_{x \in \mathcal{L}} \hat{\Psi}(x, c_j) f_k(x)}{\sum_{x \in \mathcal{L}} f_k(x)} \tag{4.5}$$

donde $\mathcal{L}$ es el conjunto de datos etiquetados de entrenamiento, $\hat{\Psi}(x, c_j) \in \{0, 1\}$ es el clasificación obtenida del mismo y $f_k(x)$ es la cantidad de veces que la característica f_k está presente en la instancia x.

Aunque las ecuaciones están bien definidas, puede suceder que el numerador de la ecuación 4.5 $\sum_{x \in \mathcal{L}} \hat{\Psi}(x, c_j) f_k(x)$ sea nulo ya que una característica puede no ocurrir en ninguna instancia de la clase. Debido a la raleza en la distribución de palabras explicada por Zipf [1935], Zipf [1949], es común que ocurra este fenómeno. Para evitarlo se suaviza los datos de la siguiente manera:

$$\hat{P}(f_k|c_j) = \frac{m_{jk} + \sum_{x \in \mathcal{L}} \hat{\Psi}(x, c_j) f_k(x)}{\sum_{x \in \mathcal{L}} (f_k(x) + m_k)} \tag{4.6}$$

Settles [2011] plantea que m_{jk} es la probabilidad anterior de f_k para la clase c_j, y comunmente se utiliza una distribución uniforme como la Laplaciana donde todos los valores son 1. El término m_k es un valor para normalizar la división.

Nuestro objetivo principal es adaptar este clasificador para que utilize un parámetro adicional m_{jk} modificando esta probabilidad anterior en base a las etiquetas del usuario para las características. Introduciremos entonces la siguiente definición:

Definición 4. La clasificación de una características es una función $\Psi_f : \mathcal{F} \times \mathcal{C} \rightarrow \{0, 1\}$ que asigna valores booleanos donde $\mathcal{F}$ es el conjunto de características y $\mathcal{C}$ es el conjunto de clases posibles.

Seguimos la implementación de dualist para agregar esta clasificación adicional de la siguiente manera:

$$m_{jk} = 1 + \Psi_f(f_k, c_j)\ \alpha \tag{4.7}$$

donde α es una constante y Ψ_f es la clasificación aprendida del usuario.

4.2. Selección de instancias

Elegir las instancias para que sean etiquetadas por el usuario es el centro de los algoritmos de aprendizaje activo y existen numerosas estrategias que pueden utilizarse. Trabajos como Settles [2009] y Schein [2005] resumen las más importantes de ellas:

Muestro por incertidumbre Asumiendo que el aprendedor tiene cierto grado de certidumbre a la hora de etiquetar un ejemplo, hay varias formas de utilizar esta información para seleccionar los ejemplos que se enviarán al oráculo:

- Confianza o incertidumbre en su clasificación.
- Distancia entre la dos primeras etiquetas más probables.
- Entropía.

Estudios han demostrado que dichas estrategias obtienen resultados similares y que la elección debe hacerse teniendo en cuenta la aplicación particular para la que se utilizarán.

Selección por comité (QBC) Se utilizan varios clasificadores para obtener un conjunto de etiquetas para cada una de las instancias no etiquetadas. Luego se seleccionan las instancias que hayan generado mayor desacuerdo entre los clasificadores.

Cambio esperado del modelo Selecciona las instancias que generarían un mayor cambio en el modelo (aprendedor) si se supiera su etiqueta. Como medida de cambio se utiliza el largo del gradiente del modelo (EGL). Se ha demostrado que funciona bien empíricamente, pero puede ser computacionalmente caro.

Reducción del error esperado Es similar al método anterior, pero en lugar de maximizar el cambio en el modelo, minimiza su error de generalización. El objetivo es reducir el número esperado de predicciones erróneas. Este método ha sido ampliamente estudiado demostrando muy buenos resultados, sin embargo, es el método computacionalmente más costoso.

Reducción de la varianza Intenta minimizar el error esperado indirectamente minimizando la varianza de los resultados. Para ello, selecciona un conjunto de instancias que maximice la información Fisher. Al igual que en el método anterior se tiene en cuenta todo es espacio de ejemplos en lugar de cada una de las instancias, y por lo tanto tiene menos probabilidades de seleccionar ejemplos raros en la distribución.

Métodos pesados por la densidad Siguiendo la misma línea que los dos métodos anteriores, supone que los ejemplos significativos son aquellos que no solo tienen alta incertidumbre, sino que son representativos de la distribución subyacente. Estudios indican resultados superiores, acompañados por implementaciones de alta velocidad que permiten incluso interacción de tiempo real con el usuario.

De todas estas opciones elegimos explorar el espacio de instancias a través de la entropía definida por Shannon [1948] como:

$$\mathcal{H}(x) = -\sum_{c_i \in \mathcal{C}} P(c_i|x)\ \log\ P(c_i|x)$$

Cover and Thomas [1991] explica en términos simples que la entropía es la cantidad de información necesaria para representar una variable aleatoria. Manning et al. [2008] plantea también que la entropía es una forma de medir la incertumbre, ya que se maximiza cuando las etiquetas para una instancia son equiprobales y se vuelve 0 cuando la instacia está etiquetada.

Este método ha sido utilizado previamente como parte del aprendizaje activo por Holub et al. [2008], Settles and Craven [2008] y Hwa [2004]. Nosotros lo tomamos por varios motivos:

- Es simple y rápido de calcular.
- Tiene en cuenta la incertidumbre de todas las clases, no solo una o dos de ellas.
- Settles [2009] sostiene que la entropía es más adecuada cuando se busca minimizar la pérdida o *log-loss*, definida como la desviación de la clasificación; mientras que los otros métodos son adecuados para minimizar el error. En particular buscamos que el clasificador se desempeñe correctamente en todas las clases y no solo en la clase mayoritaria.

4.3. Selección de características

Un criterio ampliamente utilizado para la selección de características es la Ganancia de Información o *Information Gain*. Algunos estudios que destacan su uso como medida de relevancia son Settles [2011], Forman [2003] y Sebastiani [2002]

Roobaert et al. [2006] define la ganancia de información como la cantidad de información en bits sobre la predicción de la clase, si la única información disponible es la presencia de la característica y la distribución de la clase correspondiente. En otras palabras, mide la reducción esperada de la entropía cuando la característica está presente o ausente. Podemos expresarla según la siguiente fórmula:

$$\mathcal{IG}(\mathcal{X}, \mathcal{Y}) = \mathcal{H}(\mathcal{X}) - \mathcal{H}(\mathcal{X}|\mathcal{Y}) = \mathcal{H}(\mathcal{Y}) - \mathcal{H}(\mathcal{Y}|\mathcal{X}) \tag{4.8}$$

Notemos que es una función entre variables aleatorias, donde una es la presencia de una característica y otra es la clase. Para la clasificación de texto en particular, podemos reemplazar la fórmula de la entropía y definir la ganancia de información como:

$$\mathcal{IG}(f_k) = \sum_{I_k} \sum_{j} P(I_k, c_j) \log \frac{P(I_k, c_j)}{P(I_k)P(c_j)} \tag{4.9}$$

donde $I_k \in \{0, 1\}$ indica la presencia o ausencia de la característica k.

Algunas de estas probabilidades no son parte de nuestro, por lo que las estimamos empíricamente como:

$$\hat{P}(I_k = 1) = \frac{|\{x \in \mathcal{X} : f_k(x) > 0\}|}{|\mathcal{X}|}$$

$$\hat{P}(I_k = z, c_j) = \frac{|\{x \in \mathcal{X} : I_k(x) = z \wedge \Psi(x, c_j)\}|}{|\mathcal{X}|}$$

$$\hat{P}(I_k = 0) = 1 - \hat{P}(I_k = 1)$$

donde $\mathcal{X}$ es el conjunto de instancias.

Para simplificar la tarea de etiquetamiento, presentamos al usuario las características asociadas a las clases para las que son más probables. Esta selección se realiza en dos pasos: primero seleccionamos las k características que coocurren más veces con la clase y luego las ordenamos por su ganancia de información.

4.4. Maximización de la esperanza

El algoritmo de maximización de la esperanza propuesto por Dempster et al. [1977] es ampliamente utilizado para inferir parámetros desconocidos en una distribución que tiene estados latentes. Si conocieramos el universo completo de instancias posibles y sus clases correspondientes, entonces podríamos estimar los parámetros de un clasificador directamente utilizando máxima verosimilitud. Sin embargo, no conocemos parte de este universo, lo cual constituye nuestros estados latentes.

Ya hemos definido previamente el concepto de verosimilitud o *likelihood* del modelo: es una función sobre los parámetros del modelo y un conjunto de datos que calcula la probabilidad de los valores tomados por cada variable aleatoria del modelo bajo dichos parámetros. Podemos escribirlo como:

$$L(\theta) = P(\mathcal{X}_l, \mathcal{C}_l; \theta) \tag{4.10}$$

donde θ representa a los parámetros y $\mathcal{L} = (\mathcal{X}_l, \mathcal{C}_l)$. Como las instancias de entrenamiento se asume que son i.i.d., entonces podemos escribir las dos siguientes fórmulas equivalentes:

$$L(\theta) = \prod_{x_i, c_i \in \mathcal{L}} P(x_i, c_i; \theta) \tag{4.11}$$

$$l(\theta) = \log L(\theta) = \sum_{x_i, c_i \in \mathcal{L}} \log P(x_i, c_i; \theta) \tag{4.12}$$

Si llamamos $cant(c_j)$ a la cantidad de instancias etiquetadas con la clase c_j, entonces definimos:

$$l(\theta) = \sum_{x_i \in \mathcal{L},\, c_j \in \mathcal{C}} cant(c_j) \log P(x_i, c_j; \theta) \tag{4.13}$$

$$l(\theta) = |\mathcal{X}_l| \sum_{x_i \in \mathcal{L},\, c_j \in \mathcal{C}} P(c_j) \log P(x_i, c_j; \theta) \tag{4.14}$$

A grandes rasgos, el algoritmo funciona en dos pasos E y M, por sus siglas en inglés *Expectation* y *Maximization*.

El paso E es el que calcula el valor esperado de la verosimilitud asumiendo que la distribución actual es verdadera y no existen variables no observadas. Para realizarlo, utilizamos el clasificador en su estado actual para etiquetar probabilisticamente el conjunto de datos no etiquetados, y asumimos dichas etiquetas como correctas. Con este conjunto de nuevos datos se calcula el $l(\theta)$ del modelo según la ecuación 4.14.

Si los parámetros actuales fueran correctos, entonces la verosimilitud obtenida sería la verosimilitud real del modelo, pero como no lo son provee una cota inferior. Luego, como tenemos una función sin valores no observados, el paso M maximiza $l(\theta)$ encontrando parámetros del modelo más adecuados. De esta forma optimizamos la cota inferior encontrada generando un nuevo conjunto de parámetros $\hat{\theta}$.

Ambos pasos se repite numerosas veces hasta lograr convengercia. Sin embargo tomamos ejemplo de Settles y realizamos una sola iteración, dado que argumenta que las siguientes iteraciones no aportan significativamente a la precisión del clasificador.

Si bien derivar la maximización correcta de los parámetros del $FeatMultinomialNB$ no es trivial, la implementación posterior sí es sencilla. Para realizarlo nos basamos en la maximización de un clasificador Bayesiano simple de Liu [2006], agregando la noción de Settles [2011] de utilizar tanto los datos etiquetados como no etiquetados pesando los no etiquetados por un factor de 0,1.

Al etiquetar los ejemplos no anotados obtenemos una matriz con $P(c_j|x_i)$ para cada instancia y cada clase. Utilizamos esta matriz para reestimar los parámetros de la siguiente forma:

$$\hat{P}_u(c_j) = \sum_{x_i \in \mathcal{U}} P(c_j|x_i) P(x_i)$$

$$\hat{P}_u(f_k|c_j) = \sum_{x_i \in \mathcal{U}} P(c_j|x_i) P(x_i) f_k(x_i)$$

$$\hat{P}_l(c_j) = \sum_{x_i \in \mathcal{U}} P(c_j|x_i) \Psi(x_i, c_j)$$

$$\hat{P}_l(f_k|c_j) = \sum_{x_i \in \mathcal{U}} P(c_j|x_i) \Psi(x_i, c_j) f_k(x_i)$$

$$\hat{P}(c_j) = \frac{0{,}9 * \hat{P}_l(c_j) + 0{,}1 * \hat{P}_u(c_j)}{\mathcal{Z}_1} \tag{4.15}$$

$$\hat{P}(f_k|c_j) = \frac{0{,}9 * \hat{P}_l(f_k|c_j) + 0{,}1 * \hat{P}_u(f_k|c_j)}{\mathcal{Z}_2} \tag{4.16}$$

donde $\mathcal{U}$ es el conjunto de datos no etiquetados y $\mathcal{Z}_1$, $\mathcal{Z}_2$ son contantes de normalización

Capítulo 5

Entorno de experimentación

5.1. Ejemplos seleccionados

5.1.1. Proceso previo

Al momento de extraer las caraterísticas que mencionamos en la sección anterior encontramos varior problemas. Una aproximación simple al aprendije activo incluye reentrenar el clasificador en cada una de las iteraciones del ciclo, cambiando así el modelo. Al introducir el etiquetado de características ya no se puede cambiar el modelo sin perder rastro de la ubicación de las características etiquetadas dentro de las matrices internas del clasificador. Por esto es que tuvimos que cambiar la implementación básica y extraer todos las características dentro de la instancia de preproceso. De esta forma, nuestras matrices iniciales tienen toda la información tanto del corpus anotado como no anotado.

Si bien las características anteriores pueden tomar valores en un amplio rango de enteros, decidimos sólo medir la presencia o ausencia de cada una de ellas. Por un lado, consideramos que el usuario no debería etiquetar cada característica en base a su valor, ya que este no es distintivo de la clase. Por ejemplo, si el lemma de una palabra como “movie” aparece más de 3 veces o menos no influye en las etiquetas que puedan asignarse a ellas.

5.2. Usuarios Emulados

Para poder realizar experimentos automáticos sin que el usuario tenga que ingresar la misma información repretidas veces decidimos guardar las respuestas obtenidas. Por ello agregamos etiquetas al corpus no etiquetados, que por supuesto no son consultadas, y creamos un corpus de asociaciones características-clases. El corpus de

características no es más que una matriz ternaria con tres valores posibles: asociación positiva, no asociación, desconocido. De esta forma también evitamos preguntar a un usuario por las características que ya han sido vistas pero no están relacionadas a ninguna clase en particular. Observemos que esta forma de guardar la información soporta etiquetas múltiples.

5.3. Experimentos realizados

5.3.1. Métricas utilizadas

Exactitud Llamamos exactitud a la cantidad de preguntas etiquetadas correctamente sobre el total de preguntas clasificadas.

Curva de aprendizaje Definimos la curva de aprendizaje como la exactitud del clasificador en función de la cantidad de ejemplos o características etiquetados necesarios.

Coeficiente Kappa de Cohen Esta medida ajusta la exactitud del clasificador obtenido a la de un clasificador aleatorio. Una exactitud del 80 % no es muy sorprendente si asignando etiquetas al azar obtenemos una exactitud del 70 %. En nuestro caso el corpus de evaluación contiene aproximadamente un 75 % de instacias de clase "otro", por lo tanto un clasificador que elija esta etiqueta todas las veces obtendría una exactitud semejante. Esta métrica nos permitirá superar este inconveniente y obtener valores más reales.

Una definición más formal del Coeficiente de Kappa es la que propone Carletta [1996]:

$$K = \frac{P(A) - P(E)}{1 - P(E)}$$

donde $P(A)$ es la proporción de veces que los clasificadores acuerdan y $P(E)$ es la proporción de veces que se esperaría que acuerden por casualidad. En este caso, uno de los clasificadores es el Multinomial Bayesiano entrenado y el otro son las etiquetas del corpus de evaluación. Por lo tanto, $P(A)$ no es otra cosa más que la exactitud calculada en el primer item. Adicionalmente, calculamos $P(E)$ de la siguiente forma:

$$P(E) = \frac{\sum_{i \in \mathcal{C}} Pr(\hat{x}_i) * Pr(x_i)}{|\mathcal{E}|}$$

donde $\mathcal{C}$ es el conjunton de clases, $\mathcal{E}$ es el corpus de evaluación, $Pr(\hat{x}_i)$ es la proporción de instancias etiquetadas por el clasificador con la clase i, y $Pr(x_i)$ es la proporción de instancias que pertenecen realmente a la clase i.

Precisión y exhaustividad por clase Estas dos medidas puede utilizarse sólo en clasificación binaria, por lo que tomaremos sus valores para cada una de las clases posibles. Definimos precisión como la cantidad de instancias etiquetadas para una clase que son correctas (positivos verdaderos o P_v) sobre la cantidad de instancias etiquetadas para esa clase (P_v y falsos positivos o P_f).

$$Precision(C_i) = \frac{P_v}{P_v + P_f}$$

La exhaustividad, por otro lado, está definida como la cantidad de instancias etiquetadas correctamente (P_v) de una clase dada sobre la cantidad de instancias que pertenecen a la clase verdaderamente (P_v y falsos negativos o N_f).

$$Exhaustividad(C_i) = \frac{P_v}{P_v + N_f}$$

5.3.2. Baseline

Tomaremos como baseline dos métodos:

Selección de instancias y características al azar

Los experimentos que solicitan al oráculo información sobre instancias y características se realizaron alternando pocas preguntas relativas a instancias y la misma cantidad relativas a características. Si bien los dos ciclos de etiquetado son completamente independientes en el sistema, tomamos esta desición para eliminar las diferencias entre experimentos que puedan introducirse a partir de la elección de los usuarios. Otra punto en el que difieren nuestros experimentos de una sesión no simulada con un usuario es la cantidad de veces que se reentrena. Con objeto de obtener una medición precisa de la curva de aprendizaje reentrenamos el clasificador luego de cada ronda intancias-características descriptas anteriormente. Esto es costoso para grandes volúmenes de datos y podría hacer perder al sistema su interactividad.

5.3.3. Hipótesis 1

El aprendizaje activo obtiene mejores resultados que un clasificador normal utilizando la misma cantidad de datos.

Para comprobar esta hipótesis realizamos sesiones de etiquetamiento sobre instancias y características, pero eligiendo al azar cuáles presentar al usuario. Luego compararemos las curvas de aprendizaje de ambos experimentos.

5.3.4. Hipótesis 2

El aprendizaje activo sobre instancias y características obtiene mejores resultados que el aprendizaje activo sobre instancias o características por separado.

5.3.5. Experimento 3

Hipótesis El aprendizaje supervisado sobre instancias y características obtiene mejores resultados que el aprendizaje supervisado sobre instancias, aún si las características son pocas.

5.3.6. Experimento 4

Hipótesis ??.
La idea es ver qué método de selección de features es mejor:

Puede ser trabajo futuro
Resultados

	Clase con probabilidad alta	Clase con probabilidad baja
Característica con probabilidad alta	medicion	medicion
Característica con probabilidad baja	medicion	medicion

	Clase con probabilidad alta	Clase con probabilidad baja
Característica con IG alta	medicion	medicion
Característica con IG baja	medicion	medicion

5.3.7. Experimento 5

Hipótesis Seleccionar features para etiquetar que tengan alta confiabilidad/correlación, y luego de superado un cierto límite pasar a los que tiene baja confiabilidad/correlación permite al clasificador eliminar el ruido no introducido por la baja cantidad de ejemplos y al mismo tiempo expandir la cobertura. Dejar para mas adelante

Experimento Entrenamiento supervisado con y sin etiquetado de features Validacion del etiquetado de features

Experimento 6 Information gain sobre todo el corpus o solo el etiquetado. IG sobre el corpus anotado + frecuencia en no anotado vs IG sobre todo el corpus anotado y no anotado.

Experimento 7 Coocurrencia de features con otros features. (Información mutua) Un feature se rankea mas alto si coocurre con features que se rankean alto. Tomando como base la frecuencia.

Experimento 8 Information gain sirve o alcanza sólo con usar coocurrencia?

Bibliografía

Steven Abney. *Semisupervised Learning for Computational Linguistics*. Chapman & Hall/CRC, 1st edition, 2007. ISBN 1584885599, 9781584885597.

Tim Berners-Lee. Linked data - design issues, November 2014. URL `http://www.w3.org/DesignIssues/LinkedData.html`.

Dan Brickley and Ramanathan V. Guha. Rdf vocabulary description language 1.0: Rdf schema - w3c recommendation, November 2014. URL `http://www.w3.org/TR/rdf-schema/`.

Jean Carletta. Assessing agreement on classification tasks: the kappa statistic. *Computational Linguistics*, 22(2):249–254, 1996.

Tom Heath Christian Bizer and Tim Berners-Lee. Liked data-the story so far. *International journal on semantic web and information systems*, 5(3):1–22, 2009.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, New York, NY, USA, 1991. ISBN 0-471-06259-6.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39 (1):1–38, 1977.

Gregory Druck, Burr Settles, and Andrew McCallum. Active learning by labeling features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 81–90, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-59-6. URL `http://dl.acm.org/citation.cfm?id=1699510.1699522`.

Seyda Ertekin, Jian Huang, Leon Bottou, and Lee Giles. Learning on the border: Active learning in imbalanced data classification. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 127–136. ACM, 2007. ISBN 978-1-59593-803-9.

George Forman. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, 3:1289–1305, March 2003. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944919.944974.

Eibe Frank and Remco R. Bouckaert. Naive bayes for text classification with unbalanced classes. In *Proceedings of the 10th European Conference on Principle and Practice of Knowledge Discovery in Databases*, PKDD'06, pages 503–510, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-45374-1, 978-3-540-45374-1. doi: 10.1007/11871637_49. URL http://dx.doi.org/10.1007/11871637_49.

Poonam Gupta and Vishal Gupta. A survey of text question answering techniques. *International Journal of Computer Applications*, 53(4):1–8, 2012.

A. Holub, P. Perona, and M.C. Burl. Entropy-based active learning for object recognition. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pages 1–8, June 2008. doi: 10.1109/CVPRW.2008.4563068.

TimothyM. Hospedales, Shaogang Gong, and Tao Xiang. Finding rare classes: Adapting generative and discriminative models in active learning. In JoshuaZhexue Huang, Longbing Cao, and Jaideep Srivastava, editors, *Advances in Knowledge Discovery and Data Mining*, volume 6635 of *Lecture Notes in Computer Science*, pages 296–308. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-20846-1.

Rebecca Hwa. Sample selection for statistical parsing. *Comput. Linguist.*, 30(3): 253–276, September 2004. ISSN 0891-2017. doi: 10.1162/0891201041850894. URL http://dx.doi.org/10.1162/0891201041850894.

P. Jain and A. Kapoor. Active learning for large multi-class problems. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 762–769, June 2009.

Leila Kosseim and Jamileh Yousefi. Improving the performance of question answering with semantically equivalent answer patterns. *Data Knowl. Eng.*, 66(1):53–67, 2008. ISSN 0169-023X. URL http://dx.doi.org/10.1016/j.datak.2007.07.010.

Bing Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data (Data-Centric Systems and Applications)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 3540378812.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.

Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. *AAAI-98 workshop on learning for text categorization*, 752:41–48, 1998.

Shiyan Ou and Zhenyuan Zhu. An entailment-based question answering system over semantic web data. *??*, ??(?):?–?, ??

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

Danny Roobaert, Grigoris Karakoulas, and NiteshV. Chawla. Information gain, correlation and support vector machines. In Isabelle Guyon, Masoud Nikravesh, Steve Gunn, and LotfiA. Zadeh, editors, *Feature Extraction*, volume 207 of *Studies in Fuzziness and Soft Computing*, pages 463–470. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-35487-1. doi: 10.1007/978-3-540-35488-8_23. URL `http://dx.doi.org/10.1007/978-3-540-35488-8_23`.

Andrew Ian Schein. *Active Learning for Logistic Regression*. PhD thesis, Philadelphia, PA, USA, 2005. AAI3197737.

Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, 2002. ISSN 0360-0300.

Burr Settles. Active learning literature survey. Technical Report 1648, University of Wisconsin–Madison, 2009.

Burr Settles. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. *2011 Conference on Empirical Methods in Natural Language Processing*, pages 1467–1478, 2011.

Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 1070–1079, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1613715.1613855`.

C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, July 1948.

Christina Unger, André Freitas, and Philipp Cimiano. An introduction to question answering over linked data. In Manolis Koubarakis, Giorgos Stamou, Giorgos Stoilos, Ian Horrocks, Phokion Kolaitis, Georg Lausen, and Gerhard Weikum, editors, *Reasoning Web. Reasoning on the Web in the Big Data Era*, volume 8714 of *Lecture Notes in Computer Science*, pages 100–140. Springer International Publishing, 2014. ISBN 978-3-319-10586-4.

Rui Wang and Shuguang Li. Constructing a question corpus for textual semantic relations. *??*, ??(?):?–?, ??

George Kingsley Zipf. *The psycho-biology of language: an introduction to dynamic philology.* Houghton Mifflin company, Boston, 1935.

George Kingsley Zipf. *Human behavior and the principle of least effort: An introduction to human ecology.* Hafner, 1949.