Reconocimiento de Implicación Textual Mediante Lógica

José de Jesús Lavalle Martínez Manuel Montes y Gómez
Héctor Jiménez Salazar

Reporte Técnico No. CCC-16-001
15 de enero de 2016

Luis Enrique Erro 1
Sta. Ma. Tonantzintla,
72840, Puebla, México.

Reconocimiento de Implicación Textual Mediante Lógica

José de Jesús Lavalle Martínez Manuel Montes y Gómez Héctor Jiménez Salazar

Computer Science Department
National Institute of Astrophysics, Optics and Electronics
Luis Enrique Erro # 1, Santa María Tonantzintla, Puebla, 72840, México
E-mail:{jlavalle,mmontesg}@ccc.inaoep.mx,hjimenez@correo.cua.uam.mx

Resumen

Si bien en el Procesamiento de Lenguaje Natural *(NLP) el enfoque lógico no ha sido preferido, debido principalmente al mejor desempeño que han mostrado otros enfoques [JM09], en la tarea de* Reconocimiento de Implicación Textual *(RTE) se ha vuelto a retomar [Bos13].*

En primer lugar porque la lógica es la que formaliza la noción de implicación; en segundo lugar porque, gracias a la especificación y verificación formal de software y hardware, se han desarrollado herramientas [BCM92] tanto teóricas como prácticas que han permitido razonar sobre sistemas de tamaño industrial; en tercer lugar dado que en la tarea de implicación textual ningún enfoque ha demostrado supremacía sobre los demás [KP12, AM10]; en cuarto lugar porque haciendo verificación de modelos podemos saber por qué falla la implicación textual.

Así en esta propuesta de investigación se plantea resolver la tarea de RTE usando métodos lógicos.

1. Antecedentes

En esta sección daremos el contexto en el que se ubica esta propuesta de investigación y los principales retos que tiene el Procesamiento de Lenguaje Natural.

1.1. Procesamiento de Lenguaje Natural

Para Jurafsky [JM09], el Procesamiento de Lenguaje Natural (NLP) es un campo interdisciplinario cuyo propósito es construir sistemas computacionales (agentes conversacionales o sistemas de diálogo) capaces de comunicarse con las personas en el lenguaje de éstas.

De acuerdo a esta definición, las primeras tareas que estos sistemas habrían de solucionar son *reconocimiento automático de voz* y *comprensión del lenguaje natural*, para la entrada; junto con *generación de lenguaje natural* y *síntesis de voz*, para la salida.

Una tarea que ha sido muy importante desde el incio del NLP, y que ahora recibe atención especial por toda la información disponible en la red, es la *traducción automática* de documentos entre lenguajes. Otra tarea que tiene relación directa con la Web es *responder preguntas*, lo cual permitiría realizar búsquedas en la Web haciendo preguntas completas, en lugar de sólo escribir palabras clave.

Algunas clases de preguntas como *preguntas sobre definiciones* ¿Qué significa divergente? o *preguntas sobre hechos* ¿En qué año nació Abraham Lincoln? ya las pueden responder los motores de búsqueda. No obstante, plantear preguntas más complejas requeriría extraer información de textos relacionados con una página web, implicar (sacar conclusiones sobre la base de hechos conocidos) y resumir información de muchas fuentes o páginas web.

1.2. Conocimiento Lingüístico

Por supuesto para realizar las tareas mencionadas se requiere *conocimiento lingüístico*. De este modo, para *reconocimiento de voz* y *síntesis de voz* se requiere conocimiento *fonético* y *fonológico*; para saber cómo se pronuncian las palabras de acuerdo a secuencias de sonidos, y cómo estos sonidos se producen acústicamente.

Para producir y reconocer las variaciones individuales de las palabras se requiere conocimiento *morfológico*; para saber cómo las palabras se descomponen en sus partes componentes que conllevan significado, como el caso de singular versus plural.

El orden en el que aparecen las palabras en una oración determina si la oración está bien escrita y sus posibles significados; para ello se requiere de conocimiento *sintáctico*.

Para averiguar el significado de una oración se necesita el significado de cada palabra (*semántica léxica*), también se necesita el significado de cada palabra con respecto a sus palabras adyacentes (*semántica composicional*).

Para identificar la intención (petición, afirmación o interrogación) en una oración o para distinguir si el tono de una sentencia es cortés se requiere de conocimiento *pragmático*. Finalmente, al usar pronombres, se tienen que revisar partes previas del discurso, para saber a qué se está haciendo referencia.

1.3. Ambigüedad

Se dice que una oración es ambigua si existen multiples estructuras lingüísticas alternativas que se pueden construir para ella. Los diferentes niveles de conocimiento lingüístico, vistos anteriormente, relacionan tareas de NLP con la clase de ambigüedad que resuelven.

Por ejemplo, decidir si una palabra es un verbo o un sustantivo se puede resolver mediante *etiquetado gramatical*. Decidir si un verbo tiene uno u otro sentido se puede resolver mediante *desambiguación del sentido de la palabra*, estas tareas son parte de la *desambiguación léxica*.

Saber si las palabras son parte de la misma entidad o si son entidades diferentes se resuelve mediante *desambiguación sintáctica*.

1.4. Modelos

Para construir sistemas para el NLP se han usado modelos y teorías de ciencias de la computación, matemáticas y lingüística. Entre los principales modelos están *máquinas de estados finitos*, *sistemas de reglas*, *lógica*, *modelos probabilistas* y *modelos de espacios vectoriales*. Estos modelos son implementados mediante algoritmos de *búsqueda en espacios de estados* y algoritmos de *aprendizaje computacional*.

Los modelos de máquinas de estados más usados son las versiones *deterministas* y *no deterministas* de los *autómatas de estados finitos* y *transductores de estados finitos*.

Con respecto a los sistemas de reglas, existen *gramáticas regulares*, *relaciones regulares* y *gramáticas libres de contexto*, así como variaciones probabilistas de ellas. Las máquinas de estados y los sistemas de reglas son las herramientas más usadas al tratar con conocimientos de fonología, morfología y sintaxis.

El tercer modelo que juega un papel importante al capturar conocimiento del lenguajes es el lógico. La *lógica de primer orden*, el *cálculo lambda*, el *cálculo de Lambek*, la *lógica categorial*, la *lógica modal* y la *Lógica Natural*, entre otras, se han usado. Estas representaciones lógicas se han empleado para modelar semántica y pragmática.

Los modelos *probabilistas* son cruciales para capturar toda clase de conocimiento lingüístico. La principal ventaja de los modelos probabilistas es su habilidad para resolver muchos problemas de ambigüedad, casi cualquier problema de ambigüedad se puede replantear como: "dadas N alternativas para alguna entrada ambigua, elige la más probable".

Los modelos de *espacios vectoriales*, basados en *álgebra líneal*, son el soporte para la recuperación de información y muchos tratamientos del significado de las palabras.

1.5. Definición del Problema de Investigación

De acuerdo con el Portal de Implicación Textual de la ACL, la *Implicación Textual* es una relación direccional entre dos fragmentos de texto. La relación se cumple siempre que la veracidad del segundo fragmento de texto se siga de la veracidad del primer fragmento de texto. En el marco de la Implicación Textual al texto implicante se le llama *texto* (t) y al texto implicado se le llama *hipótesis* (h).

De acuerdo a Dagan et al. [DDMR09] el reconocimiento de la implicación textual (RTE, por sus siglas en inglés) es la tarea para decidir, dados dos fragmentos de texto, si el significado del texto h es implicado (puede ser inferido) del texto t.

Ghuge y Bhattacharya [GB13] dan tres definiciones de implicación textual, a saber.

Definición clásica: Un texto t implica a la hipótesis h si h es verdadera siempre que t es verdadero, sin importar las circunstancias.

Esta defincón es muy estricta ya que requiere la veracidad de h en todos los casos (modelos) donde t es verdadero. Debido a la incertidumbre que prevalece en las aplicaciones del mundo real, esta defición no es muy útil. Por lo anterior se da la segunda definición.

Definición aplicada: Un texto t implica la hipótesis h si un humano al leer t inferirá que lo más probable es que h es verdadera.

El problema con esta definición es que es muy abstracta para ser implementada computacionalmente. Por lo tanto se da una definición matemáticamente precisa y computable en términos de probabilidades.

Definición matemática: La hipótesis h es implicada por el texto t si

$$P(h \text{ es verdadera}|t) > P(h \text{ es verdadera})$$

Para esta propuesta de investigación diremos que un texto t implica a un texto h si cada vez que un caso (modelo) satisfaga a t también satisfacerá a h. Como el título de la propuesta lo indica, se usarán sólo métodos lógicos para resolver el problema de reconocimiento de implicación textual.

La importancia de la implicación textual se basa en que captura las necesidades de inferencia semántica presentes en otras tareas del procesamiento del lenguaje natural, por ejemplo:

- En *resumen de textos* se trata de extraer las oraciones más importantes en un texto, pero se debe cuidar que las oraciones seleccionadas no impliquen la misma información, para evitar oraciones redundantes.

- En los *sistemas que responden preguntas* es importante identificar los textos que impliquen la respuesta esperada.
- En los *sistemas para extraer información* también es necesaria la implicación ya que se requiere encontrar fragmentos de texto que impliquen un patrón determinado.

Se espera que al mejorar los sistemas de implicación textual estaremos mejorando nuestro entendimiento de cómo una máquina le debe dar significado al lenguaje natural.

2. Marco Teórico

En esta sección se presentan algunas teorías lógicas que se han utilizado en el procesamiento del lenguaje natural en general y, particularmente, en el reconocimiento de implicación textual.

2.1. Teoría de Representación de Discurso

La Teoría de Representación de Discurso [KVGR11] es una de las teorías para semántica dinámica. El interés principal de estas teorías es tomar en cuenta la dependencia del contexto que tiene el significado. Es una característica ubicua de los lenguajes naturales que lo que se expresa es interpretable sólo cuando el intérprete toma en cuenta el contexto en el que se hizo, el significado de lo expresado depende del contexto.

Aún más, la interacción entre el contexto y lo expresado es recíproco. Cada expresión contribuye (vía la interpretación que se le da) al contexto en el que se hace. Modifica el contexto en un nuevo contexto, en el que esta contribución se refleja, es este nuevo contexto el que influirá en la interpretación de cualqueir cosa que se exprese posteriormente.

2.1.1. Definición

Una Estructura de Representación de Discurso [Mus96] se obtiene a partir de un conjunto de constantes de relación, un conjunto de constantes individuales y un conjunto infinito de variables individuales (a las constantes y variables individuales se les llama *referentes de discurso*). Las condiciones y las cajas se construyen a partir de estos conjuntos y de las siguientes cláusulas:

- Si R es una relación constante de aridad n y $\delta_1, \ldots, \delta_n$ son referentes de discurso entonces $R(\delta_1, \ldots, \delta_n)$ es una condición;
- Si δ_1 y δ_2 son referentes de discurso entonces δ_1 **is** δ_2 es una condición.
- Si $\gamma_1, \ldots, \gamma_m$ son condiciones ($m \geq 0$) y $x_1, \ldots, x_n$ son variables ($n \geq 0$) entonces $[x_1 \ldots x_n | \gamma_1, \ldots, \gamma_m]$ es una caja.
- Si K_1 y K_2 son cajas, entonces **not** K_1, K_1 **or** K_2 y $K_1 \Rightarrow K_2$ son condiciones;
- Si K_1 y K_2 son cajas entonces $K_1; K_2$ es una caja.

El lenguaje generado por las cláusulas anteriores se interpreta como modelos ordinarios de primer orden. Los modelos se definen como pares $< D, I >$, donde D es un conjunto arbitrario no vacío e I es una función que tiene como dominio el conjunto de constantes tal que $I(c) \in D$ para cada constante individual c e $I(R) \subseteq D^n$ para cada constante de relación R de aridad n.

Una *asignación* para tal modelo de primer orden $M =< D, I >$ es una función del conjunto de variables de referentes de discurso al dominio D. Escribimos $a[x_1, \ldots, x_n]a'$ como una abreviación de "las asignaciones a y a' difieren a lo más en sus valores para $x_1, \ldots, x_n$. Como es usual, definimos $\|\delta\|^{M,a}$ como $a(\delta)$ si δ es una variable y como $I(\delta)$ si δ es una constante.

Las cláusulas siguientes definen el valor semántico $\|\gamma\|^M$ de una condición γ en un modelo M como un conjunto de asignaciones, el valor semántico $\|K\|^M$ de una caja K en M se define como una relación binaria entre asignaciones (el super índice M se omitirá).

- $\|R(\delta_1, \ldots, \delta_n)\| = \{a \mid \langle \|\delta_1\|^a, \ldots, \|\delta_n\|^a \rangle \in I(R)\}$
- $\|\delta_1$ **is** $\delta_2\| = \{a \mid \|\delta_1\|^a = \|\delta_2\|^a\}$
- $\|[x_1 \ldots x_n|\gamma_1, \ldots, \gamma_m]\| = \{\langle a, a' \rangle \mid a[x_1, \ldots, x_n]a' \wedge a' \in \|\gamma_1\| \cap \ldots \cap \|\gamma_m\|\}$
- $\|$**not** $K\| = \{a \mid \neg \exists a' \langle a, a' \rangle \in \|K\|\}$
- $\|K_1$ **or** $K_2\| = \{a \mid \exists a' (\langle a, a' \rangle \in K_1 \vee \langle a, a' \rangle \in K_2)\}$
- $\|K_1 \Rightarrow K_2\| = \{a \mid \forall a' (\langle a, a' \rangle \in K_1 \rightarrow \exists a'' \langle a', a'' \rangle \in K_2)\}$
- $\|K_1; K_2\| = \{\langle a, a' \rangle \mid \exists a'' (\langle a, a'' \rangle \in \|K_1\| \wedge \langle a'', a' \rangle \in \|K_2\|)\}$

Una caja K es $verdadera$ en un modelo M bajo una asignación a si y sólo si existe alguna asignación a' tal que $\langle a, a' \rangle \in \|K\|^M$; una condición γ es verdadera en M bajo a si y sólo si $a \in \|\gamma\|^M$.

2.1.2. Ejemplo

Dado el siguiente texto en inglés "*A man adores a woman. She abhors him.*" La caja cerrada que le corresponde a la sentencia "*A man adores a woman.*" es (1), la única caja razonable que se puede asociar con la sentencia abierta "*She abhors him.*" es la caja abierta (2) la cual es verdadera bajo una asignación a si y sólo si la condición $x_2\ abhors\ x_1$ es verdadera bajo a.

Los pronombres anafóricos "*she*" y "*him*" obtienen cualquier valor que la asignación de entrada asocie con los referentes de discurso x_2 y x_1. La caja (2) se puede interpretar como una prueba: dada cualquier asignación de entrada a, prueba si $a(x_2)\ abhors\ a(x_1)$, si es así regresa a a como salida, si no la prueba falla y ninguna salida se regresa.

$$[x_1\ x_2|man\ x_1, woman\ x_2, x_1\ adores\ x_2] \tag{1}$$

$$[|x_2\ abhors\ x_1] \tag{2}$$

De tal manera que la caja correspondiente al texto "*A man adores a woman. She abhors him.*" es:

$$[x_1\ x_2|man\ x_1, woman\ x_2, x_1\ adores\ x_2]; [|x_2\ abhors\ x_1]$$

2.2. Gramáticas Categoriales

Las gramáticas categoriales [SB11, MR12] son una forma de gramáticas lexicalizadas, en las que la aplicación de las reglas sintácticas está condicionada completamente por el tipo sintáctico, o la *categoría* de sus entradas.

Las categorías identifican sus constituyentes como *categorías primitivas* o *funciones*. Las categorías primitivas, tales como N, NP, PP, S, etc, pueden enriquecerse con características tales como número, caso, inflección y similares.

Las funciones (tales como los verbos) portan categorías que identifican el tipo de su resultado y el de sus argumentos/complementos (ambos pueden ser a su vez funciones o categorías primitivas). Las categorías función también definen el orden en el que los argumentos se deben combinar y si ocurren a la derecha o la izquierda del funtor.

Cada categoría sintáctica se asocia con una forma lógica cuyo tipo semántico está determinado completamente por la categoría sintáctica.

En gramáticas categoriales la información sintáctica, de la clase que se puede capturar para el Inglés mediante reglas de producción como (3), (4) y (5), se transfiere a entradas léxicas como (6):

$$S \rightarrow NP \quad VP \tag{3}$$

$$VP \rightarrow TV \quad NP \tag{4}$$

$$TV \rightarrow \{proved, finds, \ldots\} \tag{5}$$

$$proved := (S\backslash NP)/NP \tag{6}$$

Esta categoría sintáctica identifica al verbo transitivo como una función, especifica el tipo y direccionalidad de sus argumentos y el tipo de su resultado. Aquí se usa la notación del "resultado más izquierdo" en la que una función que *combina a la derecha* sobre un dominio β en un rango α se escribe α/β, el correspondiente funtor que *combina a la izquierda* se escribe $\alpha\backslash\beta$, donde α y β también pueden ser categorías función.

2.2.1. Definición del Lenguaje Categorial

En Gramáticas Categoriales los *tipos* (también llamados *categorías*) se definen como sigue:

$$L ::= P|(L/L)|(L\backslash L)$$

donde P es el conjunto de tipos primitivos, los cuales son llamados tipos atómicos o categorías básicas, los tipos más usuales son S (para sentencias), NP (para frases nominales), y puede incluir PP (para frases preposicionales), INF (para infinitivos), etc.

Es usual decir que una fórmula de tipo X/Y o $X\backslash Y$ es un funtor, siendo la fórmula Y su argumento y la fórmula X su resultado.

2.2.2. Reglas de Aplicación Funcional, Categorías Sintácticas

Para permitir que funtores como (6) combinen con sus argumentos necesitamos reglas combinatorias, de éstas las dos más simples son las reglas de aplicación funcional siguientes:

$$\frac{X/Y \quad Y}{X} \; > \qquad\qquad \frac{Y \quad X\backslash Y}{X} \; <$$

2.2.3. Ejemplo de Categorías Sintácticas

Dadas la oración "*Marcel proved completeness*", y las siguientes entradas léxicas $Marcel := NP$, $proved := (S\backslash NP)/NP$ y $completeness := NP$, se procede como sigue: dado que $Marcel$ y $completeness$ son elementos del tipo primitivo NP, no se les puede aplicar alguna de las reglas de aplicación funcional, pero notamos que $proved$ tiene el tipo funcional $(S\backslash NP)/NP$ que es el tipo de los verbos transitivos.

Así, para combinar el tipo funcional $(S\backslash NP)/NP$ necesitamos a la derecha de $proved$ un elemenot del tipo NP en este caso $completeness$, usando la regla > obtenemos el tipo funcional $(S\backslash NP)$. Para combinar este tipo mediante la regla < necesitamos un elemento de NP a la izquierda de $proved$, en este caso tenemos a $Marcel$, aplicando dicha regla obtenemos el tipo S que es el tipo de las oraciones sintácticamente correctas.

El razonamiento anterior se puede expresar gráficamente mediante un árbol de derivación (también llamado de prueba) como el de la Figura 1 (al final de las líneas horizontales se marca que regla se usó para la derivación, se marca con Lex cuando se usa una entrada léxica).

$$\dfrac{\dfrac{Marcel}{NP}\ \text{Lex} \qquad \dfrac{\dfrac{proved}{(S\backslash NP)/NP}\ \text{Lex} \qquad \dfrac{completeness}{NP}\ \text{Lex}}{S\backslash NP}\ >}{S}\ <$$

Figura 1. Árbol de prueba para la oración *Marcel proved completeness***.**

2.2.4. Reglas de Aplicación Funcional, Tipos Semánticos

Se puede considerar que las categorías codifican el tipo semántico de su traducción. La traducción se puede hacer explícita asociando una forma lógica con la categoría sintáctica completa a través del operador :, se asume que éste tiene menor precedencia que los operadores categoriales / y \.

Por supuesto se deben de expandir las reglas de aplicación funcional de acuerdo a los tipos semánticos, recordando que los operadores / y \ definen categorías funcionales el resultado de aplicar las reglas > y <, cuando ya se tienen los tipos semánticos, debe ser la evaluación de una función, como se indica a continuación.

$$\dfrac{X/Y : f \quad Y : a}{X : fa}\ > \qquad\qquad \dfrac{Y : a \quad X\backslash Y : f}{X : fa}\ <$$

2.2.5. Ejemplo de Tipos Semánticos

Enriqueciendo con tipos semánticos las entradas léxicas del ejemplo "*Marcel proved completeness*" se tiene

$$\begin{aligned} Marcel &:= NP : marcel' \\ proved &:= (S\backslash NP)/NP : \lambda x.\lambda y.prove'xy \\ completeness &:= NP : completeness' \end{aligned}$$

Como se observa las categorías funcionales se enriquecen semánticamente mediante cálculo lambda y las categorías primitivas mediante una constante que se forma primando la entrada léxica, pero sin considerar su tipo sintáctico.

Aplicado la regla > para tipos semánticos a $(S\backslash NP)/NP : \lambda x.\lambda y.prove'xy$ y $NP : completeness'$ obtenemos $S\backslash NP : \lambda y.prove'completeness'y$. Si ahora le aplicamos la regla < a dicho resultado y a $NP : marcel'$, se obtiene $S : prove'completeness'marcel'$. Nuevamente este razonamiento se puede expresar mediante un árbol de derivación semántica como el de la Figura 2.

$$\dfrac{\dfrac{Marcel}{NP : marcel'}\ \text{Lex} \quad \dfrac{\dfrac{proved}{(S\backslash NP)/NP : \lambda x.\lambda y.prove'xy}\ \text{Lex} \quad \dfrac{completeness}{NP : completeness'}\ \text{Lex}}{S\backslash NP : \lambda y.prove'completeness'y}\ >}{S : prove'completeness'marcel'}\ <$$

Figura 2. Árbol de derivación semántica para la oración *Marcel proved completeness*.

2.3. Lógica Dinámica de Predicados

En DPL [Dek08] la dinámica se refiere a la información sobre cosas que se van introduciendo en un discurso y que sirven como posibles antecedentes para pronombres anafóricos subsecuentes. Como es usual en lingüística, las frases nominales (frases nominales indefinidas y pronombres) se asocian con índices o variables, para poder indicar casos de correferencia y ligado. La información relevante es información sobre los posibles valores de dichas variables, las cuales pueden ser cambiadas y actualizadas conforme avanza el discurso.

2.3.1. Definición

Se define $[\![\phi]\!]_M$, la interpretación de una fórmula ϕ de la lógica de predicados de primer orden relativa a un modelo ordinario M de la lógica de predicados de primer orden, como un conjunto de pares de asignaciones a variables, asignaciones entrada/posible-salida $\langle g, h\rangle$. La idea es que un par $\langle g, h\rangle$ está en la interpretación de ϕ relativa a M si y sólo si sobre la asignación de entrada g la fórmula ϕ puede interpretarse exitosamente y da como posible-salida la asignación h. Si no hacen falta se omitirán las referencias a M.

Un lenguaje L para DPL es el de la lógica de predicados de primer orden ordinaria, basado en un conjunto C de constantes individuales c y conjuntos R^n de constantes relacionales R de aridad n y un conjunto numerable de variables V. El conjunto de términos $T = C \cup V$ consiste de las constantes individuales y las variables del lenguaje, las fórmulas atómicas $Rt_1 \dots t_n$ se componen de predicados R de aridad n y una secuencia de n términos $t_1, \dots, t_n$, también pueden ser de la forma $t_i = t_j$, enunciando la identidad de los valores de los términos t_i y t_j. Las fórmulas se construyen a partir de las fórmulas atómicas usando negación ($\neg$), cuantificadores existencial y universal ($\exists x, \forall y$), conjunción ($\wedge$), disyunción ($\vee$) e implicación ($\rightarrow$).

Un modelo $M = \langle D, V\rangle$ es un modelo usual de primer orden con un dominio de individuos D y una función de interpretación V para las constantes individuales y relacionales de nuestro lenguaje. La función V asigna un individuo $V(c) \in D$ a las constantes individuales de L y un conjunto de $n-$tuplas de individuos $V(R^n) \subseteq D^n$ a sus constantes relacionales de aridad n. En la interpretación de DPL también usamos asignaciones variables f, g, h, k, l las cuales asignan individuos $f(x) \in D$ a las variables $x \in V$, así que son funciones de V a D. La interpretación $[t]_{M,g}$ de un término t en un modelo M y relativo a una asignación g es $V(t)$ si t es una constante individual y $g(t)$ si t es una variable.

Usamos $g[x/d]$ para la asignación variable h que es como g excepto que asigna d a x, así para toda $y \in V$, si $x \neq y$ entonces $g[x/d](y) = g(y)$ y si $x = y$ entonces $g[x/d](y) = d$. Escribimos $g[x]h$ si y sólo si $h = g[x/d]$ para algún individuo d y $g[X]h$ si y sólo si $X = \{x_1, \dots, x_n\}$ y existen $k_1, \dots, k_{n-1}$

tal que $g[x_1]k_1, \ldots, k_{n-1}[x_n]h$. Usando estos dispositivos notacionales podemos enunciar la semántica de DPL como sigue:

- $[\![Rt_1 \ldots t_n]\!]_M = \{\langle g, h\rangle \mid g = h \wedge \langle [t_1]_{M,g}, \ldots, [t_n]_{M,g}\rangle \in V(R)\}$
- $[\![t_i = t_j]\!]_M = \{\langle g, h\rangle \mid g = h \wedge [t_i]_{M,g} = [t_j]_{M,g}\}$
- $[\![\neg\phi]\!]_M = \{\langle g, h\rangle \mid g = h \wedge$ para ningún $k : \langle g, k\rangle \in [\![\phi]\!]_M\}$
- $[\![\exists x\phi]\!]_M = \{\langle g, h\rangle \mid$ para algún $k : g[x]k \wedge \langle k, h\rangle \in [\![\phi]\!]_M\}$
- $[\![\forall x\phi]\!]_M = \{\langle g, h\rangle \mid g = h \wedge$ para todo $k :$ si $g[x]k$ entonces existe $h : \langle k, h\rangle \in [\![\phi]\!]_M\}$
- $[\![\phi \wedge \psi]\!]_M = \{\langle g, h\rangle \mid$ para algún $k : \langle g, k\rangle \in [\![\phi]\!]_M \wedge \langle k, h\rangle\} \in [\![\psi]\!]_M\}$
- $[\![\phi \vee \psi]\!]_M = \{\langle g, h\rangle \mid g = h \wedge$ para algún $k : (\langle g, k\rangle \in [\![\phi]\!]_M \vee \langle g, k\rangle \in [\![\psi]\!]_M)\}$
- $[\![\phi \to \psi]\!]_M = \{\langle g, h\rangle \mid g = h \wedge$ para todo $k :$ si $\langle g, k\rangle \in [\![\phi]\!]_M$ entonces existe $h : \langle k, h\rangle \in [\![\psi]\!]_M\}$

2.3.2. Ejemplo de Interpretación Dinámica de un Discurso

Considere el siguiente texto en inglés "*A farmer owned a donkey. It was unhappy. It didn't have a tail.*", a éste le corresponde la siguiente fórmula de la Lógica Dinámica de predicados

$$\exists x(Fx \wedge \exists y(Dy \wedge Oxy)) \wedge (Uy \wedge \neg\exists z(Tz \wedge Hyz))$$

Relativo a la asignación de entrada g se tendrá un asignación de salida h si podemos encontrar asignaciones k y l tales que k es una salida posible al interpretar $\exists x(Fx \wedge \exists y(Dy \wedge Oxy))$ relativo a g, y l es una salida posible al interpretar Uy relativo a k, y h es una salida posible al interpretar $\neg\exists z(Tz \wedge Hyz)$ relativo a l.

Ya que la segunda fórmula es atómica y la tercera una negación, sabemos que en este caso $k = l$ y $l = h$. La asignación k (esto es: h) se obtiene de g al reiniciar el valor de x tal que $k(x) = h(x) \in I(F)$, y después reiniciando el valor de y tal que $k(y) = h(y) \in I(D)$ y $\langle h(x), h(y)\rangle \in I(O)$. Esto es, $h(x)$ es un granjero que posee un burro $h(y)$. Observe que para cualquier granjero f y burro d que f posee, existe una asignación correspondiente $h' : g[\{x, y\}]h'$ y tal que $h(x) = f$ y $h(y) = d$.

El segundo conyunto primero prueba si y es infeliz, esto es, si $l(y) = k(y) = h(y) \in I(U)$. El tercer conyunto, una negación, prueba si la asignación h no puede servir como entrada para satisfacer la fórmula interna $\exists z(Tz \wedge Hyz)$. Esta subfórmula se satisface relativa a h si y sólo si existe una asignación h' tal que $h[z]h'$ y $h'(z) \in I(T)$ y $\langle h'(y), h'(z)\rangle \in I(H)$, esto es, si y sólo si podemos cambiar la valuación h de z en cualquier cosa que tenga cola mediante $h(y)$.

La negación de la subfórmula prueba si no podemos cambiar la valuación de z en dicha manera. Juntando todo, $\langle g, h\rangle$ está en la interpretación de nuestro ejemplo si y sólo si $g[x, y]h$ y $h(x)$ es un granjero que posee un burro $h(y)$ el cual es infeliz y no tiene cola. Observe, una vez más, que para cualquier granjero f y burro sin cola e infeliz d al que f posee, existe una asignación correspondiente $h' : g[x, y]h'$ tal que $h(x) = f$ y $h(y) = d$.

2.4. Lógica Natural

Según van Eijck [vE07] para desarrollar un enfoque cognitivo de razonamiento es promisorio factorizar el aspecto sintáctico (aquel que tiene que ver con coincidencia de patrones y estructuras sintácticas) del resto. Un candidato obvio para esta tarea es el llamado cálculo de monotonía o cálculo de la Lógica Natural. Este cálculo tiene un lado sintáctico y un lado semántico.

El fundamento semántico del razonamiento monótono es una generalización de la noción de consecuencia lógica para tipos arbitrarios, lo cual se logra definiendo órdenes parciales $\leq$ sobre todos los tipos (no sólo el tipo de oraciones, también el de frases verbales, predicados, adjetivos, cuantificadores, etc.).

En estos términos se puede definir qué significa que una función del tipo α en el tipo β *preserve el orden* o *invierta el orden*. Las funciones que preservan el orden son las funciones f tales que si $x \leq y$ entonces $f(x) \leq f(y)$. Las funciones que invierten el orden son las funciones f tales que si $x \leq y$ entonces $f(y) \leq f(x)$.

El lado sintáctico del cálculo de monotonía tiene que ver con marcar la monotonía de las componentes de una estructura sintáctica. Sea S una estructura sintáctica, y sea A una componente de esa estructura. Suponga que A tiene tipo α y que S tiene tipo β. Considere la función sintáctica F que consiste en reemplazar la componente A por otra componente adecuada de tipo α. En otras palabras, considere la función $F = \lambda Y.S[Y/A]$. Entonces la contraparte semántica de F es una función f de tipo $\alpha \to \beta$. La validez y completitud de un cálculo de monotonía tiene que ver con la relación entre F y f.

Un algoritmo de marcación de monotonía es *válido* si se cumple lo siguiente: si A se marca + en S entonces la función que interpreta $\lambda Y.S[Y/A]$ preserva la monotonía, si A se marca − en S entonces la función que interpreta $\lambda Y.S[Y/A]$ invierte el orden.

Un algoritmo de marcación de monotonía es *completo* si se cumple lo siguiente: si la función que interpreta $\lambda Y.S[Y/A]$ preserva el orden entonces A se marca + en S, si la función que interpreta $\lambda Y.S[Y/A]$ invierte el orden entonces A se marca − en S.

2.4.1. Semántica de Monotonía

Así como podemos decir que "*Gaia is smiling*" implica lógicamente "*Gaia is smilling or Gaia is crying*", nos gustaría decir que "*smilling*" implica lógicamente "*smilling or crying*", o que "*dancing*" implica lógicamente "*moving*", también que "*at least three*" implica lógicamente "*at least two*", etc.

"*Gaia is smilling*" es una oración, "*smilling*" es un predicado, "*at least three*" es un cuantificador. Sabemos que una sentencia implica a otra si siempre que la primera es verdadera la segunda también lo es. La manera obvia de trasladar esta noción a predicados es estipulando que un predicado implica a otro si se cumple que para todo sujeto la oración que se obtiene al combinar un sujeto con el primer predicado implica a la oración que se obtiene al combinar ese mismo sujeto con el segundo predicado. Similarmente para cuantificadores, para obtener una oración a partir de "*at least three*", se tiene que combinar el cuantificador con un sustantivo y un verbo. Ya que en verdad se cumple que para todo sustantivo N y verbo V "*at least three* N V" implica "*at least two* N V", podemos decir que "*at least three*" implica "*at least two*".

Iniciaremos con los tipos básicos t (valores veritativos, el tipo de las oraciones declarativas) y e (entidades, el tipo de los nombres propios). Los tipos complejos se definen por recursión como sigue:

1. e y t son tipos,
2. si α y β son tipos, entonces (α, β) es un tipo.

Cada tipo denota a un conjunto, así D_e denota al conjunto de entidades, $D_t = \{0, 1\}$ y en general $D_{(\alpha,\beta)}$ denota al conjunto cuyos elementos son funciones de α en β.

La relación de implicación (orden parcial) sobre cada tipo se define como sigue:

1. Si $E, E' \in D_e$ entonces $E \leq_e E'$ si y sólo si $E = E'$,
2. Si $E, E' \in D_t$ entonces $E \leq_t E'$ si y sólo si $E = 0$ o $E' = 1$,
3. Si $E, E' \in D_{(\alpha,\beta)}$ entonces $E \leq_{(\alpha,\beta)} E'$ si y sólo si para todo $x \in D_\alpha, E(x) \leq_\beta E'(x)$.

2.4.2. Reglas para Razonamiento Monótono

Una función F que preserva monotonía se puede representar de la siguiente manera:

$$\frac{X \leq_\alpha Y}{F(X) \leq_\beta F(Y)} \; F\uparrow$$

Aquí se asume que X y Y son expresiones del tipo lógico α que está ordenado parcialmente mediante $\leq_\alpha$, que $F(X)$ y $F(Y)$ son expresiones del tipo β que está ordenado parcialmente mediante $\leq_\beta$, y que F es una función que preserva el orden de tipo (α, β).

Una forma de leer la regla es como una explicación del hecho de que F preserva el orden (monótona creciente). Otra manera de leer la regla es como una regla de inferencia disparada por una función F que se sabe que preserva el orden. $F\uparrow$ expresa que F preserva el orden.

Si la función F invierte el orden tenemos la siguiente regla:

$$\frac{X \leq_\alpha Y}{F(Y) \leq_\beta F(X)} \; F\downarrow$$

Nuevamente, hay varias maneras de leer esta regla. $F\downarrow$ expresa que F invierte el orden (o que es monótona decreciente).

Para apreciar la generalidad de la regla de monotonía, veamos algunos casos especiales. Si $X, Y, F(X)$ y $F(Y)$ tienen tipo t, entonces $\leq$ es consecuencia lógica (o implicación lógica), y $F(X)$ y $F(Y)$ son oraciones, de esta manera obtenemos:

$$\frac{X \leq Y \quad F(X)}{F(Y)} \; F\uparrow$$

Un ejemplo de aplicación de esta regla es: inferir de "*Mary dances implies Mary moves*" (cuando tomamos "*Mary dances*" como X y "*Mary moves*" como Y) y "*Mary dances gracefully*" (con "*gracefully*" como F) que "*Mary moves gracefully*".

Para el caso de inversión de orden tenemos:

$$\frac{X \leq Y \quad F(Y)}{F(X)} \; F\downarrow$$

Con X y Y como antes y leyendo F como negación, tenemos el siguiente ejemplo de esta regla: inferir de "*Mary dances implies Mary moves*" y "*Mary does not move*" (con "*does not move*" como F) que "*Mary does not dance*".

En el caso de que X y Y son conjuntos (con tipo (e,t)) y $F(X)$ y $F(y)$ son valores veritativos, F tiene tipo $((e,t),t)$ (el tipo de los cuantificadores), obtenemos:

$$\frac{Q(X) \quad X \subseteq Y}{Q(Y)} \; Q\uparrow$$

Como ejemplo, que X sea "*dancing*", que Y sea "*moving*" y que Q sea "*everyone*". Entonces la regla dice que podemos concluir de "*everyone is dancing*" y "*dancing involves moving*" que "*everyone is moving*".

$$\frac{Q(Y) \quad X \subseteq Y}{Q(X)} \; Q\downarrow$$

Para este caso, que X sea "*dancing*", que Y sea "*moving*" y que Q sea "*nobody*". Entonces la regla dice que podemos concluir de "*nobody is moving*" y "*dancing involves moving*" que "*nobody is dancing*".

En efecto, F puede tener más estructura interna, es decir, $F(X)$ puede tener la forma de un cuantificador generalizado binario $Quant(X,P)$ o $Quant(P,X)$. Lo cual nos da cuatro posibles reglas de monotonía para cuantificadores binarios. Ejemplos de cuantificadores binarios son: *all*, con propiedades de monotonía $(\downarrow,\uparrow)$; *some*, con propiedades $(\uparrow,\uparrow)$; *no*, con $(\downarrow,\downarrow)$ y *most*, con $(-,\uparrow)$.

$$\frac{Quant(X,P) \quad X \subseteq Y}{Quant(Y,P)} \; Quant(\uparrow,-)$$

Ejemplo, infiera de "*some philosophers are mortal*" y "*philosophers are humans*" que "*some humans are mortal*".

$$\frac{Quant(P,X) \quad X \subseteq Y}{Quant(P,Y)} \; Quant(-,\uparrow)$$

Ejemplo, infiera de "*most philosophers are human*" y "*humans are mortal*" que "*most philosophers are mortal*".

$$\frac{Quant(Y,P) \quad X \subseteq Y}{Quant(X,P)} \; Quant(\downarrow,-)$$

Ejemplo, infiera de "*all humans are mortal*" y "*philosophers are human*" que "*all philosophers are mortal*".

$$\frac{Quant(P,Y) \quad X \subseteq Y}{Quant(P,X)} \; Quant(-,\downarrow)$$

Ejemplo, infiera de "*no philosophers are mortal*" y "*humans are mortal*" que "*no philosophers are human*".

2.4.3. Polaridad de una Oración

Ahora veremos el algoritmo de Sánchez Valencia (citado y explicado por Dowty [Dow94]) para obtener la polaridad de una oración.

Hay que tomar en cuenta que Sánchez Valencia utiliza una versión de gramáticas categoriales llamada *Cálculo de Lambek* para el análisis sintáctico de la oración, pero usa la versión donde los tipos funcionales (funtores) no están dirigidos[1] y se construyen mediante el operador "," (vease la definición de tipos en 2.4.1).

[1] A diferencia de lo que vimos en la sección 2.2.1 donde los funtores se construyen mediante los operadores / y \, el primero busca su argumento a la derecha y el segundo a la izquierda.

En el cálculo de Lambek la única regla de inferencia toma la forma:

$$\frac{(\alpha,\beta) \qquad \alpha}{\beta}$$

y no importa si el tipo α aparece a la izquierda o la derecha del funtor (α,β).

Como ejemplo en la Figura 3 tenemos un árbol de prueba para la oración *Sue caught every armadillo*. En la figura note que: el tipo de *every* tiene a la derecha su argumento, el tipo de *every armadillo* tiene su argumento a la izquierda, el tipo de *Sue* tiene su argumento a la derecha y que se ha derivado el tipo t indicando que la expresión en lenguaje natural es una oración.

$$\dfrac{\dfrac{Sue}{((e,t),t)} \qquad \dfrac{\dfrac{caught}{(e,(e,t))} \qquad \dfrac{\dfrac{every}{((e,t),((e,(e,t)),(e,t)))} \qquad \dfrac{armadillo}{(e,t)}}{((e,(e,t)),(e,t))}}{(e,t)}}{t}$$

Figura 3. Árbol de prueba para la oración *Sue caught every armadillo***.**

Para encontrar la polaridad de una oración, Sánchez Valencia define un método consistente en tres pasos: marcación léxica de monotonía, marcación externa de monotonía y determinación de la polaridad de cada constituyente.

Para la marcación léxica de la monotonía considera dos aspectos: primero, asume que las palabras de tipo funtor tendrán la marca "+" cuando dicho funtor preserva el orden (monótono creciente) y la marca "–" cuando invierte el orden (monótono decreciente); segundo, formaliza el punto anterior introduciendo la siguiente regla de formación de tipos.

Si (α,β) es un tipo, entonces (α^+,β) y (α^-,β) también son tipos.

La categoría plana (α,β) sigue siendo una categoría al igual que las otras dos, representa en el contexto de monotonía a las funciones no monótonas.

Como ejemplos de marcación léxica de monotonía tenemos los siguientes:

a(n) es asignada al tipo $((e,t)^+,((e,t)^+,t))$,

every es asignada al tipo $((e,t)^-,((e,t)^+,t))$,

no es asignada al tipo $((e,t)^-,((e,t)^-,t))$.

La Figura 4 nos muestra el árbol de prueba para *Sue caught every armadillo* con marcas léxicas.

Para la marcación externa de monotonía agrega "+" y "–" a los nodos funtor y argumento en un árbol de prueba, bajo las siguientes reglas:

$$\frac{(\alpha,\beta) \qquad \alpha}{\beta} \implies \frac{\overset{+}{(\alpha,\beta)} \qquad \alpha}{\beta},$$

$$\frac{(\alpha^+,\beta) \qquad \alpha}{\beta} \implies \frac{\overset{+}{(\alpha^+,\beta)} \qquad \overset{+}{\alpha}}{\beta},$$

$$\frac{(\alpha^-,\beta) \qquad \alpha}{\beta} \implies \frac{\overset{+}{(\alpha^-,\beta)} \qquad \overset{-}{\alpha}}{\beta}$$

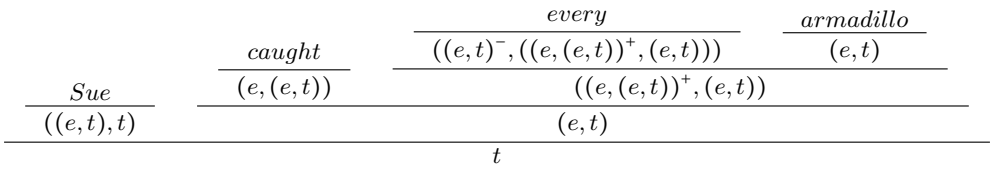

Figura 4. Árbol de prueba con marcación léxica de monotonía para la oración *Sue caught every armadillo***.**

En las Figuras 5 y 6 se muestran árboles de prueba con marcación externa de monotonía, para la oración *Sue caught every armadillo* y para la oración *Sam didn't catch every armadillo*.

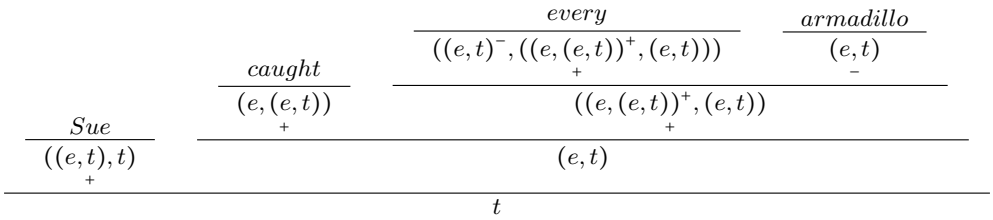

Figura 5. Árbol de prueba con marcación externa de monotonía para la oración *Sue caught every armadillo***.**

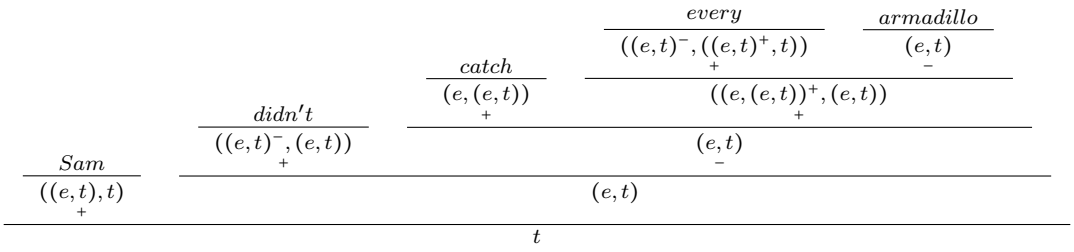

Figura 6. Árbol de prueba con marcación externa de monotonía para la oración *Sam didn't catch every armadillo***.**

Note que todavía no se puede determinar la dirección de las inferencias, ya que no son suficientes la marcación léxica y la marcación externa, también depende del contexto en que aparecen las palabras.

Por ejemplo, suponga que $female\ armadillo \leq_{(e,t)} armadillo$, que el tipo de $armadillo$ no tiene marcación léxica de monotonía, y que $armadillo$ tiene marcación externa de monotonía negativa, tanto en la Figura 5 como en la Figura 6, tenemos que:

- De *Sue caught every armadillo* **podemos inferir** que *Sue caught every female armadillo*, pero
- De *Sam didn't catch every armadillo* **no podemos inferir** que *Sam didn't catch every female armadillo*.

Por ello, para determinar la dirección real de inferencia (creciente ↑, decreciente ↓, o ninguna), se tiene la noción de *polaridad* de una oración y de sus constituyentes.

Si D es un árbol de derivación sintáctica con nodo raíz ρ, entonces:

1. Un nodo γ tiene polaridad en D si y sólo si todos los nodos en la ruta de γ a ρ están marcados;
2. un nodo γ es *positivo* si y sólo si γ tiene polaridad y el número de nodos en la ruta a la raíz marcados con "–" es par;
3. un nodo γ es *negativo* si y sólo si γ tiene polaridad y el número de nodos en la ruta a la raíz marcados con "–" es impar'.

Una forma de mostrar un resumen del cálculo de la polaridad de los constituyentes de una oración es poniendo un super índice +(−) a cada palabra (los nodos hoja de un árbol de prueba) si su polaridad es positiva(negativa).

Por ejemplo en la Figura 5 la polaridad del nodo hoja correspondiente a *armadillo* es negativa ya que tiene polaridad y en la ruta a la raíz el número de nodos marcados con − es uno, por lo tanto se le pone un super índice –, $armadillo^-$; para la misma figura *every* es marcado $every^+$ ya que tiene polaridad y en la ruta a la raíz el número de nodos marcados con − es cero.

En la figura 6 la polaridad del nodo hoja correspondiente a *armadillo* es postiva ya que tiene polaridad y en la ruta a la raíz el número de nodos marcados con − es dos, por lo tanto se le pone un super índice +, $armadillo^+$; para la misma figura *every* es marcado $every^-$ ya que tiene polaridad y en la ruta a la raíz el número de nodos marcados con − es uno.

Como los nodos internos son el resultado de aplicar una función a su argumento se pone un super índice +(−) al resultado de dicha aplicación si su polaridad es positiva(negativa). El nodo raíz no tiene polaridad por lo cual no se marca.

Por ejemplo en la Figura 5 la polaridad del nodo correspondiente a la aplicación de la función *every* a su argumento *armadillo* es positiva ya que tiene polaridad y el número de nodos en la ruta a la raíz marcados con "–" es cero, por lo tanto se marca $every^+(armadillo^-)^+$.

En el caso de la Figura 6 la polaridad del nodo correspondiente a la aplicación de la función *every* a su argumento *armadillo* es negativa ya que tiene polaridad y el número de nodos en la ruta a la raíz marcados con "–" es uno, por lo tanto se marca $every^-(armadillo^+)^-$.

Los resúmenes completos de polaridad para las oraciones *Sue caught every armadillo* y *Sam didn't catch every armadillo* son (vea Figuras 5 y 6):

$$Sue^+(caught^+(every^+(armadillo^-)^+)^+)$$
$$Sam^+(didn't^+(catch^-(every^-(armadillo^+)^-)^-)^+)$$

La Figura 7 contiene un árbol de prueba con marcación externa de polaridad para la oración *An armadillo danced* que será utilizado en la sección 8.1, su respectivo resumen de polaridad es

$$An^+(armadillo^+)^+(danced^+)^+$$

.

Una vez que se tiene la polaridad de los constituyentes de la oración se tienen las siguientes reglas de inferencia: Si la expresión M tiene polaridad positiva en la derivación $N(M)$:

$$\frac{[\![M]\!] \leq [\![M']\!] \qquad N(M)}{N(M')}$$

$$\dfrac{\dfrac{\dfrac{An}{((e,t)^{+},((e,t)^{+},t))} \quad \dfrac{armadillo}{(e,t)}}{((e,t)^{+},t)} \quad \dfrac{danced}{(e,t)}}{t}$$

Figura 7. Árbol de prueba con marcación externa de monotonía para la oración *An armadillo danced***.**

Si la expresión M' tiene polaridad negativa en la derivación $N(M')$:

$$\frac{[\![M]\!] \leq [\![M']\!] \qquad N(M')}{N(M)}$$

Como ejemplo de aplicación de la regla de inferencia para polaridad positiva tenemos:

$$\frac{[\![armadillo]\!] \leq [\![animal]\!] \qquad Sam^{+}(didn't^{+}(catch^{-}(every^{-}(armadillo^{+})^{-})^{-})^{+})}{Sam^{+}(didn't^{+}(catch^{-}(every^{-}(animal^{+})^{-})^{-})^{+})}$$

Para ejemplificar el funcionamiento de la regla de inferencia para polaridad negativa considere:

$$\frac{[\![female\ armadillo]\!] \leq [\![armadillo]\!] \qquad Sue^{+}(caught^{+}(every^{+}(armadillo^{-})^{+})^{+})}{Sue^{+}(caught^{+}(every^{+}(female\ armadillo^{-})^{+})^{+})}$$

3. Trabajos Relacionados

En esta sección se presentan los enfoques con los que se ha tratado el problema de reconocimiento de implicación textual, se comentan muy brevemente los enfoques que no usan algún tipo de lógica, con más detalle aquellos que usan a la lógica en alguna fase del proceso de reconocimiento y se finaliza con un análisis de los enfoques que usan lógica.

3.1. Enfoques que no Usan Lógica

Para Dagan et al. [DDMR09, DRSZ13] el contar con los conjuntos de datos para los retos en RTE ha permitido formular el problema en términos de clasificación, las características se obtienen de los ejemplos de entrenamiento y luego se usan los algoritmos de aprendizaje computacional para construir un clasificador, éste se aplica a los datos de prueba para clasificar cada par (t, h) como positivo o negativo. Las características pueden ser léxico-sintácticas y semánticas, basadas en el conteo de palabras y reglas de reescritura sintáctica de primer orden, para extraer la ganancia de información se usan medidas léxicas.

Otra línea trata de derivar la hipótesis h a partir del texto t por medio de un número de técnicas basadas en transformaciones sobre la representación sintáctica de t y h.

Por supuesto el enfoque probabilístico no podía faltar, en Ghuge y Bhattacharya [GB13] se menciona el cálculo de dependencias en los árboles sintácticos y la estimación de parámetros basados en la Maximización de la Esperanza.

Rus et al. [RMMG08] proponen una solución basada en grafos. Los dos fragmentos de texto t y h son representados mediante un grafo obteniendo un grafo-t y un grafo-h, si el grafo-t subsume al grafo-h, entonces t implica h, en otro caso no se da la implicación.

3.2. Enfoques que sí Usan Lógica

Aquí veremos trabajos que usan algún tipo de lógica para resolver el problema de implicación textual.

3.2.1. Hodges, Clarck, Fowler y Moldovan

El sistema de Hodges et al. [HCFM06] convierte el texto t y la hipótesis h a forma lógica, para dicha conversión requieren marcar partes de la oración, generar árboles sintácticos, desambiguar el sentido de las palabras y detectar relaciones semánticas.

Una vez que se tienen las formas lógicas de t y h utilizan COGEX (un demostrador automático de teoremas que es una versión de OTTER adaptado al Procesamiento de Lenguaje Natural). El demostrador requiere una lista de cláusulas llamadas "conjunto de apoyo" para iniciar la busqueda de inferencias. El conjunto de apoyo se carga con las forma lógica negada de la hipóteis y la forma lógica del texto.

También se requiere una segunda lista, llamada la "lista usable". La lista usable contiene axiomas que proporcionan conocimiento del mundo externo (310 axiomas), conocimiento sobre la equivalencia sintáctica entre predicados en forma lógica (reglas de reescritura lingüística) y conocimiento en forma de cadenas léxicas (cadena de relaciones entre dos *synsets* de WordNet).

Una vez que se terminan de crear el conjunto de apoyo y la lista de usables, el demostrador empieza a buscar pruebas. Las cláusulas en la lista de apoyo son ponderadas en el orden en que deben ser elegidas para participar en la búsqueda.

A la hipótesis negada se le asigna la mayor ponderación para asegurar que será la última cláusula en participar en una prueba. El demostrador quita del conjunto de apoyo la cláusula con menor ponderación y busca en la lista de usables para ver si se pueden hacer nuevas inferencias. A las inferencias que se van produciendo se les asigna una ponderación acorde con los axiomas de los que son derivadas y se anexan al conjunto de apoyo.

El demostrador continua de esta manera hasta que el conjunto de apoyo se vacía. Si se encuentra una refutación entonces la prueba está completa. Si no se encuentra una refutación se relajan los argumentos de los predicados. Si la relajación de argumentos falla en encontrar una refutación se quitan predicados de la hipótesis negada hasta que se encuentre una refutación.

Cuado se encuentra una prueba por refutación se calcula una puntuación para dicha prueba, se empieza con una puntuación perfecta y se descuentan puntos por cada axioma que se usó, por los argumentos que se relajaron y los predicados que se quitaron. Las puntuaciones se normalizan calculando la pena máxima que se le puede dar a un par al quitar todos los predicados de la hipótesis.

Debido a la técnicas de relajación siempre se encuentra una prueba, por ello para determinar si hay implicación se examinan las penas asignadas por el demostrador. Entre más axiomas se utilicen y más predicados se quiten es mucho menos probable la implicación; todas las puntuaciones por debajo de un umbral son consideradas como implicaciones falsas, las que son mayores que el umbral se consideran implicaciones verdaderas. El umbral se calcula examinando las puntuaciones de salida del conjunto de datos de desarrollo para determinar que umbral produce la exactitud mayor (.63). Seleccionando umbrales específicos a cada tarea[2] logran una exactitud de .563 con el conjunto de datos de prueba de RTE-2.

[2]En los retos RTE-1 al RTE-5 los conjuntos de datos provienen de siete tareas diferentes de procesamiento de lenguaje natural

3.2.2. Akhmatova

Akhmatova [Akh05] representa el significado de una oración como un conjunto de proposiciones atómicas contenidas en ella y compara las proposiciones para comparar las oraciones.

Para descomponer una oración en sus proposiciones atómicas usa un análisis semántico dirigido por sintaxis, la implementación del método usa la salida del analizador sintáctico como entrada para el analizador semántico, a su vez éste produce una salida de la cual se puede derivar una representación en lógica de primer orden. A la representación final de signficado le llaman *fórmula lógica de la oración*.

Argumenta que existen muchas enfoques para describir el significado por medio de una forma lógica, como un ejemplo, la oración *A restaurante serves meat* puede tener la descripción:

$$exists\ e, x\ Isa(e, Serving) \& \ Server(e, x)$$
$$\& \ Served(e, Meat)\ \&\ Isa(x, Restaurant),$$

pero es rígida y difícil de producir, por lo cual propone una representación simplificada.

Define tres tipos de objetos $Subj(x)$, $Obj(x)$ y $Pred(x)$, un elemento de asignación de significado $iq(x, < meaning\ of\ x >)$, dos variantes de relaciones $attr(x, y)$ y $prep(x, y)$. Con esta notación se puede representar sinonimia semántica ($iq(x, serve) \leftrightarrow iq(x, dish)$), hiperonimia ($iq(x, serve) \rightarrow iq(x, provide)$) y reglas de relaciones léxicas ($all\ x(iq(x, is) \leftrightarrow iq(x, be))$).

Usa un algoritmo para encontrar relaciones en WordNet, lo que el algoritmo entrega es una puntuación de la relación que tienen dos palabras, tanto para la relación de sinonimia como para la de hiperonimia. La puntuación se calcula tomando la distancia que hay entre los sentidos de las palabras en el grafo, entre más grande sea la ruta menor es la relación de las palabras.

Para decidir si hay implicación toman el siguiente criterio, si para toda proposición en la hipótesis existe una en el texto de la que pueda ser implicada (via OTTER) entonces la implicación entre t y h se cumple, de otra manera la implicación no se cumple.

Reporta una exactitud de 0.5188 con un recuerdo de 0.1025 para RTE-1, no menciona si sobre el conjunto de datos o sobre el conjunto de desarrollo.

3.2.3. Bayer, Burguer, Ferro, Henderson y Yeh

Bayer et al. [BBF+05] trabajan en MITRE corporation, enviaron dos sistemas al reto RTE-1, el segundo inspirado en modelos estadísticos de traducción automática, el primero comentado aquí brevemente.

El primer sistema procesa el texto y la hipótesis usando un analizador léxico y segmentador de oraciones construido en MITRE, el marcador de partes de la oración de Ratnaparkhi, el analizador morfológico Morph de la Universidad de Sussex, el analizador Link Grammar de CMU y el analizador de dependencias y generador lógico Davidsoniano construido en MITRE.

Después de dicho preprocesamiento el texto y la hipótesis se comparan usando EPILOG, el motor de inferencia probabilística orientado a eventos de la Universidad de Rochester.

Muy poco conocimiento semántico adicional es utilizado, sólo se agregan unas pocas reglas de inferencia y listas simples de palabras para clasificación semántica. Los autores reconocen que gracias a la pobreza de la base de conocimiento el sistema no puede probar la implicación para virtualmente todos los datos de RTE-1. Reportan una exactitud de 0.52 y un recuerdo de 0.04.

[DRSZ13].

3.2.4. Raina, Ng y Manning

Para Reina et al. [RNM05] el primer paso al procesar una oración es construir un grafo de dependencias sintácticas. La oración se analiza utilizando un analizador sintáctico. Reglas escritas a mano se usan para encontrar las cabezas de todos los nodos en el árbol de análisis sintáctico.

Este procedimiento implícitamente descubre relaciones sintácticas entre las palabras de la oración, ya que cuando una palabra se elige como la cabeza, sus hermanas deben ser modificadores sintácticos de esa palabra. Estas relaciones se pueden representar como un *grafo de dependencias* donde cada nodo es una palabra/frase y las ligas representan dependencias particulares.

Después traducen las relaciones representadas en el grafo de dependencias en una representación de fórmula lógica. Cada nodo en el grafo se convierte en un término lógico y se le asigna una constante única. Las aristas en el grafo son representadas por argumentos compartidos entre los nodos y reciben argumentos de todos los nodos ligados.

Para realizar una demostración automática estricta no se deben hacer suposiciones para realizar la prueba. El método de refutación-resolución efectua la demostración al agregar la negación de la fórmula lógica meta a una base de conocimiento que consiste de axiomas dados, luego deriva una cláusula *null* por medio de sucesivos pasos de resolución.

La anterior estrategia considera unificar un cláusula de término simple y el primer término de otra cláusula, lo cual es muy restrictivo. Sería bueno "unificar" términos como $purchase(C, A, B)$ y $\neg bougth(Z, X, Y)$. Por lo que usan abducción ponderada que permite que pares como el anterior unifiquen con algún costo calculado.

Considere los término lógicos $S(s_1, s_2, \ldots, s_m)$ y $\neg T(t_1, t_2, \ldots, t_n)$ donde S y T son predicados y $s_{1\ldots m}, t_{1\ldots n}$ son los argumentos (variables o constantes). La definición estándar de unificación requiere que $S = T$, $m = n$ y cada s_i debe unificar consistentemente con t_i. Por lo que relajan la definición estándar de unificación permitiendo que los predicados S y T sean diferentes, que el número de argumentos m y n sean distintos, que s_i pueda unificar con culquier s_j y que un argumento constante unifique con cualquier otro siempre y cuando represente la misma entidad.

Cada una de las relajaciones anteriores se interpreta como una suposición abductiva sobre el mundo, y su grado de plausibilidad se cuantifica mediante un costo no negativo mediante el *modelo de costo de suposiciones*. El modelo de costo es el responsable de abordar la semántica del lenguaje.

Dado el modelo de costo, la demostración abductiva de teoremas puede enmarcarse como un problema de búsqueda. Ya que para cada par de términos puede resultar muchos resolventes y a cada resolvente se le asigna un costo no negativo mediante el modelo de costo. Una prueba está completa cuando se alcanza la cláusula nula a través de pasos sucesivos; el costo total de la prueba es la suma de los costos de los pasos individuales.

La intuición detrás es que entre más relajaciones se hacen el costo incrementa y la implicación es menos admisible. Por ello se trata de encontrar la prueba con el costo mínimo. El modelo de costo usa un vector de ponderaciones el cual es elegido automáticamente mediante un algoritmo de aprendizaje.

Sus resultados para RTE-1 son $.578$ de exactitud para el conjunto de desarrollo y $.555$ para el conjunto de prueba, cuando usan el mismo umbral para todas las tareas. Una exactitud de $.561$ para el conjunto de desarrollo y $.57$ para el conjunto de prueba cuando entrenaron un umbral para cada tarea.

3.2.5. da Salvo Braz, Girju, Punyakanok, Roth and Sammons

En el enfoque de da Salvo Braz et al. [dSBGP+05] se utilizan los siguientes componentes:

KR: Una representación de conocimiento jerárquico basada en Lógica Descriptiva de Características Extendidas, en la que representan el texto a nivel superficial, aumentado con analizadores sintáctico y semántico inducidos y abstracciones a nivel de palabras y frases.

KB: Una base de conocimiento que consiste de reglas de reescritura sintáctica y semántica. Cada regla se escribe como $lhs \sqsubseteq rhs$ y describe la relación de subsunción entre dos representaciones, lhs (el cuerpo de la regla) y rhs (la cabeza de la regla).

Subsunción: Un algoritmo de subsunción extendido que determina la subsunción entre dos representaciones. Por *extendido* se entiende que el operador de unificación básico se extiende para que apoye varias abstracciones a nivel de palabras y frases.

El proceso inicia con un conjunto de recursos basados en aprendizaje computacional usados para inducir las representaciones para t y h. El algoritmo de implicación procede en dos fases: primero, genera incrementalmente representaciones nuevas a partir de la representación superficial original de t, aumentándola con las cabezas de las reglas de reescritura subsumidas; segundo, usan un algoritmo de subsunción extendido (basado en optimización) para ver si alguna de las representaciones alternativas de t implica la representación de h.

El algoritmo de subsunción extendida se usa tanto para verificar si hay implicación como para determinar cuándo y cómo generar una representación nueva de maneras ligeramente diferentes. El problema de subsunción lo solucionan formulando y resolviendo un problema equivalente de programación lineal entera.

Obtienen una exactitud de .659 para el conjunto de datos de prueba del RTE-1.

3.2.6. Clark y Harrison

Clark y Harrison [CH09] describen el funcionamiento de su sistema BLUE, de la siguiente manera.

BLUE consiste de dos módulos de implicación en línea. El primero genera y compara una representación lógica (una estructura de árbol simplificada y normalizada con elementos de tipo lógico, generada por reglas paralelas a las reglas gramaticales) de t y h para tratar de concluir si hay implicación o contradicción, si no lo logra el segundo módulo hace una comparación similar pero sólo usando bolsa de palabras, ignora la estructura sintáctica. WordNet y DIRT se pueden usar en ambos módulos.

Para ver si la representación de la oración h subsume a (es más general que, o es implicada por) t, BLUE usa WordNet para reconocer las relaciones de equivalencia (sinonimia) y subsunción (hiperonimia) entre los sentidos de las palabras.

Además de comparar la forma lógica de t y h, BLUE busca elaboraciones de t que sean subsumidas por h aplicandole a t reglas de inferencia. Se aplica una regla si la condición de la regla subsume a la oración h, en cuyo caso la conclusión de la regla es agregada después de ligar las variables compartidas. Su fuente de reglas de inferencia (paráfrasis) es la base de datos de reglas de inferencia DIRT.

Para el conjunto de datos de prueba de RTE-5[3] obtienen .6 de exactitud usando sólo el módulo de bolsa de palabras, .567 usando sólo el módulo lógico y .615 usando el sistema completo, para la prueba de dos vías.

Usando únicamente el módulo de bolsa de palabras obtienen .528, solamente el módulo lógico .463 y con el sistema completo .547, para la prueba de tres vías.

[3] A partir del reto RTE-5 se tiene la posibilidad de contestar sí/no (dos vías) hay implicación y sí/no/se-desconoce (tres vías) hay implicación.

3.2.7. Bos y Markert

Bos y Markert [BM06a, BM06b] hacen dos tipos de análisis semántico: superficial y profundo.

3.2.7.1. Análisis Semántico Superficial

En muchos ejemplos de implicación textual existe una dependencia entre la similitud superficial de los pares (t, h) y la existencia de la propia implicación. En particular, la inclusión de todas o casi todas las palabras de la hipótesis en el texto resulta en que la implicación es probable.

En contraste, la ocurrencia de palabras en la hipótesis que no están relacionadas con alguna palabra en el texto hace improbable la implicación.

Por lo que Bos y Markert usan el modelo de *bolsa de palabras* para medir el traslape de palabras. El procedimiento es como sigue:

1. Se encuentran las entidades léxicas y lemas del texto y la hipótesis.
2. La medida de traslape `wnoverlap` entre el texto y la hipótesis se inicia a cero.
3. En caso de que un lema en la hipótesis esté relacionado con un lema en el texto, su *ponderación* se agrega a la medida de traslape, en otro caso se ignora.
4. Al final el traslape se normaliza dividiéndolo entre la suma de las ponderaciones de todos los lemas en la hipótesis. Esto asegura que el traslape siempre es un número real entre 0 y 1, también asegura que sea independiente de la longitud de la hipótesis.

Utilizan WordNet como fuente de conocimiento para sinonimia y derivaciones. De tal manera que un lema l_1 en la hipótesis está *relacionado* con un lema l_2 en el texto si y sólo si l_1 y l_2 son iguales, pertenecen al mismo *synset* de WordNet, están relacionados vía derivaciones de WordNet o están relacionados vía una combinación de sinónimos y derivaciones. No se hace desambiguación del sentido de las palabras y se consideran *todos* los *synsets* de cada palabra.

Con respecto al asignamiento de ponderaciones, a cada lema en la hipótesis se le asigna como su ponderación la inversa de la frecuencia en el documento, utilizan la Web como corpus vía el GoogleAPI.

Además de `wnoverlap` también consideran como características la longitud del texto, la longitud de la hipótesis y la longitud relativa de la hipótesis con respecto al texto.

3.2.7.2. Análisis Semántico Profundo

Para el análisis semántico profundo usan un analizador sintáctico de cobertura amplia basado en Gramáticas Categoriales Combinatorias para generar una representación semántica de grano fino para cada par (t, h).

El lenguaje de representación semántica es un fragmento de primer orden del lenguaje de Estructuras de Representación de Discurso (DRS) que se usa en la Teoría de Representación de Discurso (DRT) [KVGR11].

Emplean la traducción estándar de DRS a Lógica de Primer Orden propuesta por Kamp y Reyle [KVGR11], con el fin de mapear las representaciones semánticas en el formato requerido por las herramientas de inferencia.

Para cada par (t, h) calculan el posible conocimiento del entorno (bk) a través de dos fuentes: relaciones de hiponimia tomadas de WordNet y un conjunto de reglas de inferencia codificadas manualmente que expresan conocimiento general.

Para verificar si una implicación se cumple o no, usan dos técnicas de razonamiento automático: demostración automática de teoremas de primer orden y verificación de modelos finitos; las cuales ejecutan las siguientes inferencias:

1. Prueba $t \rightarrow h$
2. Prueba $(bk \wedge t) \rightarrow h$
3. Prueba $\neg(bk \wedge t)$
4. Prueba $\neg(bk \wedge t \wedge h)$
5. Satisface $bk \wedge t$
6. Satisface $(bk \wedge t \wedge h)$

En 1 y 2 se verifica directamente si hay implicación entre t y h: en 1 sin conocimiento del entorno, en 2 con conocimiento del entorno. (Note que si se encuentra una prueba para 1, siempre habrá una prueba para 2.)

Los casos 3 y 4 se usan para verificar la consistencia del conocimiento del entorno con t y con t y h, respectivamente. Algunas veces la combinación del conocimiento del entorno con el texto o la hipótesis causa una inconsistencia lógica. Esto se puede deber a errores en la interface entre sintaxis y semántica, o a errores en la generación de conocimiento relevante del entorno. En ambos casos, si se encuentra una prueba, entonces el conocimiento del entorno es inconsistente. (Note que si hay una prueba para 3 también se sigue lógicamente que existe una prueba para 4.)

Finalmente, en 5 y 6 se buscan modelos de primer orden con conocimiento del entorno para t y para t y h, respectivamente. Lo cual es posible sólo si no se encuentran pruebas para 3 y 4.

3.2.7.3. Clasificación de la Implicación

Además de las cuatro características obtenidas en el análisis semántico superficial (ver 3.2.7.1), del demostrador automático de teoremas obtienen dos características: `entailed` que determina si t implica h e `inconsistent` si t junto con h son inconsistentes.

De la verificación de modelos extraen seis características: `domainsize` y `modelsize` para $t + h$, asi como las diferencias absolutas y relativas entre los tamaños de t y $t+h$, tanto para el tamaño de los dominios (`domainsizeabsdif` y `domainsizereldif`) y el tamaño de los modelos (`modelsizeabsdif` y `modelsizereldif`).

Expresan cada par (t, h) del conjunto de entrenamiento mediante un vector de características y entrenan un árbol de decisión que clasifica la implicación en TRUE o FALSE, para lo cual usan la herramienta para aprendizaje computacional Weka.

Reportan que enviaron dos corridas al segundo reto de implicación textual (RTE-2): para la primera utilizaron sólo las características tomadas del análisis semántico superficial (ver 3.2.7.1) y lograron una exactitud del .616; en el caso de la segunda utilizaron las características tanto del análisis semántico superficial como del análisis semántico profundo (ver 3.2.7.2), en este caso la exactitud fue del .606, y reportan un recuerdo bajo (no dan el porcentaje preciso).

3.2.8. MacCartney y Manning

MacCartney y Manning [MM07, Mac09] utilizan Lógica Natural para el reconocimiento de la implicación textual. La Lógica Natural evita la notación lógica y la teoría de modelos. A su sistema le llaman *Natlog*.

relación	símbolo	en términos de $\leq$
equivalente	$t = h$	$t \leq h, h \leq t$
adelante	$t \sqsubset h$	$t \leq h, h \not\leq t$
reversa	$t \sqsupset h$	$h \leq t, t \not\leq h$
independiente	$t \# h$	$t \not\leq h, h \not\leq t$
exclusiva	$t \mid h$	$t \leq \neg h$

Cuadro 1. Las cinco relaciones elementales de implicación.

Sus pruebas se desarrollan editando incrementalmente las expresiones en lenguaje natural y sus reglas de inferencia especifican las condiciones bajo las cuales las expansiones (producidas por las funciones que preservan el orden) o contracciones (producidas por las funciones que invierten el orden) semánticas preservan la veracidad.

Lo anterior permite precisar el razonamiento sobre monotonicidad, evitando las dificultades para traducir las oraciones a lógica de primer orden.

NatLog tiene una arquitectura de tres etapas, las cuales son: 1. Preprocesamiento lingüístico, 2. alineamiento y 3. clasificación de implicación.

3.2.8.1. Preprocesamiento Lingüístico

Para el preprocesamiento lingüístico usan el analizador sintáctico de Stanford (un analizador sintáctico estadístico entrenado con *Treebank*) para identificar los elementos léxicos, etiquetar partes del discurso y hacer un análisis sintáctico basado en gramáticas de estructura de frase.

El análisis más importante realizado en esta etapa es la marcación de monotonicidad [vE07] para cada entidad léxica y tramo de cada oración de entrada.

3.2.8.2. Alineamiento

La segunda etapa es un alineamiento entre el texto y la hipótesis. En Natlog los alineamientos se representan como secuencias de *ediciones atómicas* sobre tramos de entidades léxicas.

Se definen cuatro tipos de ediciones atómicas: *borrar* un tramo del texto, *insertar* un tramo en la hipótesis, *sustituir* un tramo de la hipótesis por un tramo del texto y *avanzar* sobre un tramo sin hacer modificaciones.

3.2.8.3. Clasificación de la Implicación

La secuencia de edición obtenida durante la etapa de alineamiento descompone efectivamente el problema de implicación global en una secuencia de problemas de implicación entre átomos y predice una relación de implicación para cada edición atómica.

Posteriormente se componen las predicciones de implicación atómica para producir una predicción de implicación global.

El modelo de implicación atómica usa un clasificador para predecir una de cinco relaciones elementales de implicación (ver Cuadro 1) para cada edición atómica.

El modelo de implicación usa un clasificador mediante árboles de decisión entrenado con un conjunto pequeño de datos, 69 problemas diseñados para ejercitarlo en las regiones diversas del espacio de características.

Primer Autor	Mecanismo de Inferencia	Lógica	BK	Reto	Exactitud	Decisión vía
Hodges	COGEX	LPO	WordNet	RTE-2	.63	Optimización
Akhmatova	OTTER	LPO	WordNet	RTE-1	.5188	
Bayer	EPILOG	LPO		RTE-1	.52	
Raina		LPO		RTE-1	.57	Algoritmo de Aprendizaje
da Salvo		Lógica Descriptiva	WordNet	RTE-1	.659	Optimización
Clark			WordNet y DIRT	RTE-5	.615	Subsunción
Bos	Vampire y Paradox	LPO	WordNet	RTE-1	.606	Árbol de Decisión
MacCartney		Lógica Natural	WordNet	RTE-3	.5738	Árbol de Decisión

Cuadro 2. Resumen de las principales características de los métodos que usan lógica.

Usan su sistema NatLog con el banco de pruebas FraCaS, éste está dividido en nueve secciones: 1. Cuantificadores, 2. Plurales, 3. Anáfora, 4. Elipsis, 5. Adjetivos, 6. Compartivos, 7. Temporalidad, 8. Verbos y 9. Actitudes. Por las características de NatLog, relativizan sus resultados contabilizando solamente las secciones 1, 5 y 6, obteniendo .76 de exactitud (por esto obtienen buenos resultados en recuerdo y malos en exactitud).

Para los conjuntos de datos del RTE-3 reportan una exactitud del $.58$ para el conjunto de desarrollo y $.5738$ para el conjunto de prueba.

3.3. Análisis de los Enfoques Lógicos

Dentro de los enfoque que usan lógica pudimos apreciar en 3.2 que lo hacen mediante subconjuntos de Lógica de Primer Orden, una extensión de Lógica Descriptiva y Lógica Natural; de los resultados comparables, los del reto RTE-1, los resultados están en el rango $[.5188, .659]$ con una diferencia de $.1402$, es decir, muy poca diferencia en exactitud a pesar de las diferencias en cuanto a formas lógicas usadas, ver Cuadro 2.

También se pudo observar que no es el mecanismo de inferencia (COGEX, OTTER, EPILOG, Vampire o Paradox) el que determina si t implica h, la decisión se toma optimizando un funcional, mediante un algoritmo de aprendizaje o mediante árboles de decisión.

Seis de las ocho propuestas usan WordNet para obtener *conocimiento del entorno ("background knowledge")*, pero Bos [Bos13] y Dagan et al. [DRSZ13] arguyen que uno de los problemas principales en el reconocimiento de implicación textual es justamente la adquisición de conocimiento del entorno, es por ello que los resultados en RTE son muy parecidos, independientemente del enfoque usado.

Dentro de las propuestas llaman la atención la de Bos y Markert [BM06b, BM06a, Bos13] y la de MacCartney y Manning [MM07]. La de Bos y Markert porque usan Estructuras de Representación de Discurso y hay una traducción estándar de dichas estructuras a Lógica de Primer Orden, además de que dichas estructuras tratan con el fenómeno de Anáfora[4] (de hecho para eso fueron creadas). Dicho fenómeno lingüístico es el que más presencia tiene en RTE [Sam11].

La de MacCartney y Manning [MM07] porque hacen una implementación de Lógica Natural, la cual fue creada para tratar directamente con oraciones en lenguaje natural, sólo que en lugar de determinar la polaridad de las oraciones (ver 2.4.3) las alinean, por ello en lugar de definir qué significa que una oración implique directamente a otra (ver 8.1), tienen que entrenar un árbol de decisión con las secuencia de ediciones para el alineamiento de los datos de desarrollo.

Ambas propuestas tienen ventajas y desventajas: Bos y Markert logran buena exactitud pero mal recuerdo; MacCartney y Manning, por el contrario, logran buen recuerdo pero mala exactitud.

[4]Se define la Anáfora como el fenómeno lingüístico de apuntar hacia atrás a un ítem mencionado previamente en el texto. La palabra o frase que apunta hacia atrás se llama elemento anafórico y la entidad a la que refiere es su antecedente [Mit03].

Desde un punto de vista más técnico los diversos métodos usados tienen también ventajas y desventajas: la DRT tiene como ventaja que fue creada para solucionar los problemas de Anáfora que las gramáticas de Montague no podían manejar [KVGR11], pero tiene el inconveniente de que su salida es una fórmula de la lógica de primer orden, por tanto razonar sobre su validez es indecidible en general.

Las CG [SB11] tienen la ventaja de que sintaxis y semántica están íntimamente ligadas y que la semántica se encuentra evaluando una expresión del cálculo lambda, su principal desventaja es que no pueden manejar el fenómeno de la Anáfora.

La Lógica de Predicados Dinámica (DPL) [Dek08] fue creada inicialmente para razonar sobre los cambios de estado que tiene un programa imperativo, posteriormente se ha utilizado para tratar la Anáfora. Como se demuestra en [GS91], la Lógica de Primer Orden y la Lógica de Predicados Dinámica son equivalentes, así que (DPL) tiene la misma desventaja que DRT.

Por último, la Lógica Natural [vE07] tiene la desventaja de que no puede tratar con el fenómeno de la Anáfora (entre otros) creemos que por ello es baja en exactitud, su ventaja es que es una teoría decidible.

4. Preguntas de Investigación

Como afirman MacCartney y Manning [MM07], en la Lógica Natural se evita la notación lógica y la teoría de modelos, pero como hemos visto en 2.1 y 2.3 es muy importante la noción lógica de modelo para poder dotar a un formalismo con comportamiento dinámico (en el primer caso a las estructuras de representación de discurso y en el segundo a la lógica de primer orden). De aquí que la primera pregunta de investigación se plantea como: ¿Es posible construir una teoría de modelos para la Lógica Natural?

Si se logra definir una teoría de modelos para la Lógica Natural entonces la segunda pregunta la podemos enunciar de la siguiente manera: ¿Es posible fusionar la Lógica Natural con la Lógica Dinámica?

El fenómeno lingüístico de Anáfora tiene varias formas: anáfora pronominal, anáfora de frase nominal léxica, anáfora de sustantivo, anáfora de verbo, anáfora de adverbio y anáfora cero [Mit02]. Una vez que la Lógica Natural tenga un comportamiento dinámico y por lo tanto pueda tratar con el fenómeno lingüístico de Anáfora, la tercera pregunta es: ¿Qué tipos de anáfora, además de la pronominal, pueden ser resueltos con la fusión de Lógica Natural con Lógica Dinámica?

Una de las ventajas que tienen los algoritmos de verificación de modelos [BK08], con respecto a los demostradores automáticos de teoremas, es que trabajan sobre la negación de la expresión que se quiere demostrar y buscan una interpretación que no satisfaga a la expresión, si la encuentran entonces la expresión no siempre es verdadera, pero entonces se sabe exactamente bajo que asignación de variables, constantes y símbolos predicados no se satisface la expresión.

En términos de especificación y verificación formal de sistemas, se sabe exactamente bajo qué circunstancias el sistema tendrá un comportamiento no deseado. En términos de reconocimiento de implicación textual se sabría qué relación entre objetos evita (cuando se verifica lo contrario) o no permite (cuando se desconoce si la relación entre objetos se da o no) establecer la implicación. Esto nos lleva a la cuarta pregunta: ¿Es posible definir un algoritmo de verificación de modelos para la fusión entre Lógica Natural y Lógica Dinámica?

5. Objetivos

5.1. General

Fusionar Lógica Natural con Lógica Dinámica para resolver en términos estrictamente lógicos el problema de implicación textual.

5.2. Específicos

- Crear una teoría de modelos para Lógica Natural.
- Dar comportamiento dinámico a la Lógica Natural para manejar el fenómeno lingüístico de Anáfora.
- Hacer verificación de modelos para la fusión entre Lógica Natural y Lógica Dinámica que permita decidir si la oración t implica a la oración h.

6. Contribuciones Esperadas

- Una teoría de modelos para la Lógica Natural.
- La fusión de Lógica Natural con Lógica Dinámica.
- Hacer verificación de modelos para la fusión de Lógica Natural con Lógica Dinámica.
- Una herramienta computacional para el reconocimiento de implicación textual basado exclusivamente en lógica.

7. Metodología

1. Definir una Teoría de Modelos para Lógica Natural (TMLN): asignándole un conjunto dominio a cada tipo de una gramática categorial, definiendo como se comporta la aplicación de funtores, marcando la polaridad de cada subexpresión de una oración dada y definiendo una relación de implicación ($\models$) entre un par de oraciones.
2. Demostrar que la TMLN es válida y completa con respecto a la Teoría de Pruebas para Lógica Natural de Sánchez-Valencia: primero, hay que garantizar que cada vez que se deriva un tipo Z en la Teoría de Pruebas de Sánchez-Valencia la función de significado de la TMLN se evalua a D_Z y viceversa; segundo, hay que demostrar que cada vez que el algoritmo externo de polaridad de Sánchez-Valencia marca un nodo con polaridad positiva(negativa), la función de significado con marcación interna de polaridad marca el mismo nodo con polaridad positiva(negativa) y viceversa.
3. Definir y programar un verificador de modelos para la Lógica Natural: una vez que se ha definido la relación de implicación entre un par de oraciones ($t \models h$), se puede programar en un lenguaje declarativo (por ejemplo ML[5]), para esto hay que modificar la Definición 8.3 para que incluya marcación interna de polaridad. Para reducir el espacio de estados se pueden hacer cambios sólo sobre las subexpresiones de t que tengan polaridad positiva (dualmente, sólo sobre las subexpresiones de h que tengan polaridad negativa).

[5]ML es un lenguaje de programación funcional, de alto orden y fuertemente tipificado creado por Robin Milner. Robert Harper tiene disponible en línea un buen libro para programar en ML `http://www.cs.cmu.edu/~rwh/smlbook/book.pdf`.

4. Identificar diferentes tipos de Anáfora: buscar similitudes entre las diferentes formas de anáfora de tal manera que se puedan adaptar al esquema de pares de asignaciones visto en 2.1 y 2.3.

5. Definir elementos dinámicos sobre la TMLN: incluir en la función de significado con marcación interna de polaridad la noción de par de asignaciones y cómo va a operar dicho par para lograr ligar un elemento anafórico con su antecedente.

6. Definir y programar un verificador de modelos para Lógica Natural-Dinámica: la Definición 8.7 de implicación entre un par de oraciones ($t \vDash h$), no va a cambiar lo que sí cambia y hay que reprogramar es la Definición 8.3 para que incluya la manera en que operan los pares de asignaciones.

7. Seleccionar del conjunto de datos del RTE1 aquellos pares de oraciones que contengan palabras cuyos tipos sean monótonos y que tengan elementos anafóricos.

8. Ejecutar el verificador de modelos sobre los pares seleccionados y hacer un análisis cualitativo de su comportamiento.

8. Resultados Preliminares

El estudio de algunos de los formalismos que se han usado para darle significado al lenguaje natural, sobre todo alrededor de la implicación textual, dio origen a una publicación [LMMyGJS$^+$14].

Con respecto a una Teoría de Modelos para Lógica Natural en 8.1 presentamos una primera propuesta, la definición 8.1 formaliza en el estilo de teoría de modelos el significado que se le da a los elementos del Lenguaje Categorial (ver 2.2.1), la definición 8.3 formaliza el significado que se le debe dar en teoría de modelos a la aplicación de funcional en categorías sintácticas (ver 2.2.2).

Mediante los teoremas 8.1 y 8.2 hemos demostrado que nuestra teoría de modelos es equivalente al sistema deductivo de la versión de gramáticas categoriales presentada en este documento.

La definición 8.4 permite caracterizar precisamente la noción de subexpresión en términos de nuestra teoría de modelos, en la biblografía consultada dicha noción se entendía simplemente por el contexto.

Una vez que sabemos que es una subexpresión podemos precisar también como al conectar los conceptos de polaridad y orden parcial obtenemos el concepto de implicación lineal en un paso, definición 8.5.

Aprovechar que las relaciones de orden parcial son transitivas nos permite formalizar cómo se hacen implicaciones en más de un paso, sobre una subexpresión fija, definición 8.6.

Cambiar en cualquier momento la subexpresión sobre la que hacemos implicaciones enriquece, en general, las posibilidades de inferencia de la Lógica Natural; en particular, permite precisar cómo se puede usar la Lógica Natural en el reconocimiento de implicación textual, sin que se tenga que recurrir a la noción de alineamiento y a entrenar árboles de decisión para resolver si hay implicación textual (ver 3.2.8), por ello es muy importante la definición 8.7.

Con estos elementos proponemos una arquitectura de nuestro sistema de reconocimiento de implicación textual en 8.2.

8.1. Teoría de Modelos para LN

Necesitamos definir qué es un modelo $\mathcal{M} = (D, [\![]\!])$ para Lógica Natural, donde D es un conjunto no vacío llamado *dominio de discurso* y $[\![]\!]$ es la *función de significado* que mapea las expresiones de un lenguaje en elementos y/o conjuntos del dominio de discurso, en esta sección desarrollaremos una primera propuesta de teoría de modelos para Lógica Natural.

Como se vio en la sección marco teórico, se le da significado a las expresiones de un lenguaje de acuerdo a sus reglas de construcción. El "problema" con la Lógica Natural es que sus expresiones son precisamente oraciones en lenguaje natural, fue diseñada para prescindir de algun tipo de forma lógica para representar oraciones en lenguaje natural.

Con gramáticas categoriales podemos hacer el análisis sintáctico de una oración en lenguaje natural, por ello vamos a usar estas gramáticas para construir una teoría de modelos para Lógica Natural. Si una oración en lenguje natural está bien construida, podemos derivar el tipo t a partir de la composición de los tipos de cada palabra que contine la oración.

Así, un modelo para la Lógica Natural no consiste de un sólo dominio, sino de una familia de dominios, empezando con el dominio D_e correspondiente a los elementos de tipo e (de los nombres de entidades), luego el dominio D_t correspondiente a los elementos de tipo valores de verdad, es decir, $D_t = \{0, 1\}$, y una familia de dominios $D_{X \to Y}$ correspondiente a los funtores con dominio X y rango Y.

Recordemos que en gramáticas categoriales los tipos se construyen mediante $L ::= P|(L/L)|(L\backslash L)$, donde P es el conjunto de tipos primitivos (en nuestro caso e y t) y para construir funtores usamos los operadores / y \, por lo que la función de significado es:

Definición 8.1. Sean $e, t \in P; X, Y \in L$ entonces

- $[\![e]\!] = D_e$,
- $[\![t]\!] = D_t$,
- $[\![(X/Y)]\!] = [\![(X\backslash Y)]\!] = D_{Y \to X}$ si $[\![Y]\!] = D_Y$ y $[\![X]\!] = D_X$.

□

Así tenemos en general que si $\alpha \in L$ entonces $[\![\alpha]\!] = D_\alpha$, como notación si $x \in D_\alpha$ diremos que x tiene tipo α (en símbolos $x : \alpha$).

Como todas las palabras constituyentes de una expresión tienen un tipo queremos poder darles significado como una secuencia de tipos, así tenemos la siguiente definición.

Definición 8.2. Sean $X, Y \in L$ entonces $[\![X]\!][\![Y]\!] = [\![XY]\!]$.

□

Como dijimos para probar (demostrar) que una expresión en lenguaje natural está bien construida (es una oración) se construye un árbol de prueba cuya raíz sea t, utlilizando las reglas de aplicación funcional para categorías sintácticas $\dfrac{X/Y \quad Y}{X}$ y $\dfrac{Y \quad X\backslash Y}{X}$, como ejemplo tenemos el árbol de prueba de la Figura 8.

$$\dfrac{\dfrac{Sue}{(t/(t/e))} \qquad \dfrac{\dfrac{caught}{((t/e)/e)} \qquad \dfrac{\dfrac{every}{(((t/e)\backslash((t/e)/e))/(t/e))} \qquad \dfrac{armadillo}{(t/e)}}{((t/e)\backslash((t/e)/e))}}{(t/e)}}{t}$$

Figura 8. Árbol de prueba para la oración *Sue caught every armadillo***.**

Para que en nuestra teoría de modelos podamos hacer un análisis equivalente tenemos que darle significado a la aplicación de funtores, así tenemos la siguiente definición.

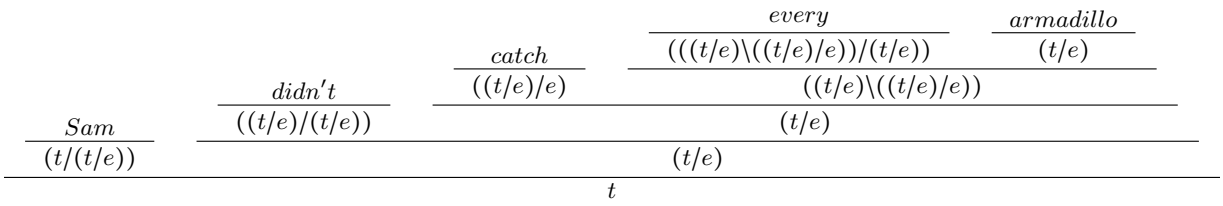

Figura 9. Árbol de prueba para la oración *Sam didn´t catch every armadillo*.

Definición 8.3. Sean $X, Y \in L$, Z_1 y Z_2 secuencias de tipos posiblemente vacías, entonces

$$[\![Z_1\ (X/Y)\ Y\ Z_2]\!] = [\![Z_1\ Y\ (X\backslash Y)\ Z_2]\!] = [\![Z_1\ X\ Z_2]\!].$$

□

Ejemplo 8.1. Vamos a tomar los tipos de las palabras de la oración *Sue caught every armadillo* y calcularemos su significado mediante las Definiciones 8.1 y 8.3, para ver si podemos reducirlo al dominio D_t.

$$[\![\overbrace{(t/(t/e))\quad((t\backslash e)/e)}^{Z_1}\quad\overbrace{(((t/e)\backslash((t\backslash e)/e))/(t/e))}^{(X/Y)}\quad\overbrace{(t/e)}^{Y}]\!] \overset{8.3}{=}$$

$$[\![\overbrace{(t/(t/e))}^{Z_1}\quad\overbrace{((t\backslash e)/e)}^{Y}\quad\overbrace{((t/e)\backslash((t\backslash e)/e))}^{(X\backslash Y)}]\!] \overset{8.3}{=}$$

$$[\![\overbrace{(t/(t/e))}^{(X/Y)}\quad\overbrace{(t/e)}^{Y}]\!] \overset{8.3}{=}$$

$$[\![t]\!] \overset{8.1}{=} D_t.$$

□

Ahora demostraremos en general que si podemos construir un árbol de prueba con raíz $Z \in L$, entonces Z tiene significado.

Teorema 8.1. Si $Z \in L$ es la raíz de un árbol de prueba entonces $[\![Z]\!] = D_Z$.
Demostración por inducción sobre la profundidad del árbol de prueba.
Si la profundidad del árbol de prueba es 0 tenemos los siguientes casos:

1. $Z = e$ por lo tanto $[\![e]\!] \overset{8.1}{=} D_e$,
2. $Z = t$ por lo tanto $[\![t]\!] \overset{8.1}{=} D_t$,
3. $Z = (X/Y)$ por lo tanto $[\![(X/Y)]\!] \overset{8.1}{=} D_{Y\to X}$ si $[\![Y]\!] = D_Y$ y $[\![X]\!] = D_X$,
4. $Z = (X\backslash Y)$ por lo tanto $[\![(X\backslash Y)]\!] \overset{8.1}{=} D_{Y\to X}$ si $[\![Y]\!] = D_Y$ y $[\![X]\!] = D_X$.

Si la profundidad del árbol de prueba es n tenemos los siguientes casos:

1. $Z = e$ por lo tanto es la raíz de uno de los dos árboles siguientes:

a) $\dfrac{(e/\overset{\vdots}{Y}) \quad \overset{\vdots}{Y}}{e}$ entonces (e/Y) y Y son las raices de árboles de prueba de profundidad menor que n, así por hipótesis de inducción tenemos que $[\![(e/Y)]\!]\ [\![Y]\!] \overset{8.2}{=} [\![(e/Y)\ Y]\!] \overset{8.3}{=} [\![e]\!] \overset{8.1}{=} D_e$,

b) $\dfrac{\overset{\vdots}{Y} \quad (e\backslash\overset{\vdots}{Y})}{e}$ entonces Y y $(e\backslash Y)$ son las raices de árboles de prueba de profundidad menor que n, así por hipótesis de inducción tenemos que $[\![Y]\!]\ [\![(e\backslash Y)]\!] \overset{8.2}{=} [\![Y\ (e\backslash Y)]\!] \overset{8.3}{=} [\![e]\!] \overset{8.1}{=} D_e$.

2. $Z = t$ por lo tanto es la raíz de uno de los dos árboles siguientes:

a) $\dfrac{(\overset{\vdots}{t}/Y) \quad \overset{\vdots}{Y}}{t}$ entonces (t/Y) y Y son las raices de árboles de prueba de profundidad menor que n, así por hipótesis de inducción tenemos que $[\![(t/Y)]\!]\ [\![Y]\!] \overset{8.2}{=} [\![(t/Y)\ Y]\!] \overset{8.3}{=} [\![t]\!] \overset{8.1}{=} D_t$,

b) $\dfrac{\overset{\vdots}{Y} \quad (\overset{\vdots}{t}\backslash Y)}{t}$ entonces Y y $(t\backslash Y)$ son las raices de árboles de prueba de profundidad menor que n, así por hipótesis de inducción tenemos que $[\![Y]\!]\ [\![(t\backslash Y)]\!] \overset{8.2}{=} [\![Y\ (t\backslash Y)]\!] \overset{8.3}{=} [\![t]\!] \overset{8.1}{=} D_t$.

3. $Z = (Z_1/Z_2)$ por lo tanto es la raíz de uno de los dos árboles siguientes:

a) $\dfrac{((Z_1/\overset{\vdots}{Z_2})/Z_3) \quad \overset{\vdots}{Z_3}}{(Z_1/Z_2)}$ entonces $((Z_1/Z_2)/Z_3)$ y Z_3 son las raices de árboles de prueba de profundidad menor que n, así por hipótesis de inducción tenemos que $[\![((Z_1/Z_2)/Z_3)]\!]\ [\![Z_3]\!] \overset{8.2}{=} [\![((Z_1/Z_2)/Z_3)\ Z_3]\!] \overset{8.3}{=} [\![(Z_1/Z_2)]\!] \overset{8.1}{=} D_{Z_2 \to Z_1}$,

b) $\dfrac{\overset{\vdots}{Z_3} \quad ((Z_1/\overset{\vdots}{Z_2})\backslash Z_3)}{(Z_1/Z_2)}$ entonces Z_3 y $((Z_1/Z_2)\backslash Z_3)$ son las raices de árboles de prueba de profundidad menor que n, así por hipótesis de inducción tenemos que $[\![Z_3]\!]\ [\![((Z_1/Z_2)\backslash Z_3)]\!] \overset{8.2}{=} [\![Z_3\ ((Z_1/Z_2)\backslash Z_3)]\!] \overset{8.3}{=} [\![(Z_1/Z_2)]\!] \overset{8.1}{=} D_{Z_2 \to Z_1}$.

4. $Z = (Z_1\backslash Z_2)$ por lo tanto es la raíz de uno de los dos árboles siguientes:

a) $\dfrac{((Z_1\backslash\overset{\vdots}{Z_2})/Z_3) \quad \overset{\vdots}{Z_3}}{(Z_1\backslash Z_2)}$ entonces $((Z_1\backslash Z_2)/Z_3)$ y Z_3 son las raices de árboles de prueba de profundidad menor que n, así por hipótesis de inducción tenemos que $[\![((Z_1\backslash Z_2)/Z_3)]\!]\ [\![Z_3]\!] \overset{8.2}{=} [\![((Z_1\backslash Z_2)/Z_3)\ Z_3]\!] \overset{8.3}{=} [\![(Z_1\backslash Z_2)]\!] \overset{8.1}{=} D_{Z_2 \to Z_1}$,

b) $\dfrac{\overset{\vdots}{Z_3} \quad ((Z_1\backslash\overset{\vdots}{Z_2})\backslash Z_3)}{(Z_1/Z_2)}$ entonces Z_3 y $((Z_1\backslash Z_2)\backslash Z_3)$ son las raices de árboles de prueba de profundidad menor que n, así por hipótesis de inducción tenemos que $[\![Z_3]\!]\ [\![((Z_1\backslash Z_2)\backslash Z_3)]\!] \overset{8.2}{=} [\![Z_3\ ((Z_1\backslash Z_2)\backslash Z_3)]\!] \overset{8.3}{=} [\![(Z_1\backslash Z_2)]\!] \overset{8.1}{=} D_{Z_2 \to Z_1}$.

□

El siguiente teorema prueba que si $Z \in L$ tiene significado entonces hay un árbol de prueba cuya raíz es Z.

Teorema 8.2. Sea $Z \in L$, sí $[\![Z]\!] = D_Z$ entonces existe un árbol de prueba cuya raíz es Z.
Demostración por inducción sobre el número de veces que se utilizó la definición 8.3 para obtener D_Z.

1. Si la definición 8.3 se utilizó 0 veces tenemos cuatro casos:
   a) $[\![e]\!] \overset{8.1}{=} D_e$ entonces le corresponde el árbol $\dfrac{}{e}$, es decir, un árbol de profundidad 0 con raíz e,
   b) $[\![t]\!] \overset{8.1}{=} D_t$ entonces le corresponde el árbol $\dfrac{}{t}$, es decir, un árbol de profundidad 0 con raíz t,
   c) $[\![(X/Y)]\!] \overset{8.1}{=} D_{Y\to X}$ entonces le corresponde el árbol $\dfrac{}{(X/Y)}$, es decir, un árbol de profundidad 0 con raíz (X/Y),
   d) $[\![(X\backslash Y)]\!] \overset{8.1}{=} D_{Y\to X}$ entonces le corresponde el árbol $\dfrac{}{(X\backslash Y)}$, es decir, un árbol de profundidad 0 con raíz $(X\backslash Y)$.
2. Si la definición 8.3 se utilizó n veces tenemos ocho casos:
   a) $D_e \overset{8.1}{=} [\![e]\!] \overset{8.3}{=} [\![(e/Y)\ Y]\!] \overset{8.2}{=} [\![(e/Y)]\!]\ [\![Y]\!]$, pero para obtener $[\![(e/Y)]\!]$ y $[\![Y]\!]$ la definición 8.3 se utilizó menos de n veces, así por hipótesis de inducción existen árboles de prueba de los cuales son raices, a saber $\dfrac{\vdots\ (e/Y)\quad \vdots\ Y}{}$, ahora aplicando la regla de aplicación funcional para categorías sintácticas obtenemos el árbol $\dfrac{\vdots\ (e/Y)\quad \vdots\ Y}{e}$,
   b) $D_e \overset{8.1}{=} [\![e]\!] \overset{8.3}{=} [\![Y\ (e\backslash Y)]\!] \overset{8.2}{=} [\![Y]\!]\ [\![(e\backslash Y)]\!]$, pero para obtener $[\![Y]\!]$ y $[\![(e\backslash Y)]\!]$ la definición 8.3 se utilizó menos de n veces, así por hipótesis de inducción existen árboles de prueba de los cuales son raices, a saber $\dfrac{\vdots\ Y\quad \vdots\ (e\backslash Y)}{}$, ahora aplicando la regla de aplicación funcional para categorías sintácticas obtenemos el árbol $\dfrac{\vdots\ Y\quad \vdots\ (e\backslash Y)}{e}$,
   c) $D_t \overset{8.1}{=} [\![t]\!] \overset{8.3}{=} [\![(t/Y)\ Y]\!] \overset{8.2}{=} [\![(t/Y)]\!]\ [\![Y]\!]$, pero para obtener $[\![(t/Y)]\!]$ y $[\![Y]\!]$ la definición 8.3 se utilizó menos de n veces, así por hipótesis de inducción existen árboles de prueba de los cuales son raices, a saber $\dfrac{\vdots\ (t/Y)\quad \vdots\ Y}{}$, ahora aplicando la regla de aplicación funcional para categorías sintácticas obtenemos el árbol $\dfrac{\vdots\ (t/Y)\quad \vdots\ Y}{t}$,
   d) $D_t \overset{8.1}{=} [\![t]\!] \overset{8.3}{=} [\![Y\ (t\backslash Y)]\!] \overset{8.2}{=} [\![Y]\!]\ [\![(t\backslash Y)]\!]$, pero para obtener $[\![Y]\!]$ y $[\![(t\backslash Y)]\!]$ la definición 8.3 se utilizó menos de n veces, así por hipótesis de inducción existen árboles de prueba de los cuales son raices, a saber $\dfrac{\vdots\ Y\quad \vdots\ (t\backslash Y)}{}$, ahora aplicando la regla de aplicación funcional para categorías sintácticas obtenemos el árbol $\dfrac{\vdots\ Y\quad \vdots\ (t\backslash Y)}{t}$,
   e) $D_{Z_2\to Z_1} \overset{8.1}{=} [\![(Z_1/Z_2)]\!] \overset{8.3}{=} [\![((Z_1/Z_2)/Z_3)\ Z_3]\!] \overset{8.2}{=} [\![((Z_1/Z_2)/Z_3)]\!]\ [\![Z_3]\!]$, pero para obtener $[\![((Z_1/Z_2)/Z_3)]\!]$ y $[\![Z_3]\!]$ la definición 8.3 se utilizó menos de n veces, así por hipótesis de in-

ducción existen árboles de prueba de los cuales son raices, a saber $\dfrac{((Z_1/\overset{\vdots}{Z_2})/Z_3) \quad \overset{\vdots}{Z_3}}{}$, ahora aplicando la regla de aplicación funcional para categorías sintácticas obtenemos el árbol $\dfrac{((Z_1/\overset{\vdots}{Z_2})/Z_3) \quad \overset{\vdots}{Z_3}}{(Z_1/Z_2)}$,

f) $D_{Z_2 \to Z_1} \overset{8.1}{=} [\![(Z_1/Z_2)]\!] \overset{8.3}{=} [\![Z_3\ ((Z_1/Z_2)\backslash Z_3)]\!] \overset{8.2}{=} [\![Z_3]\!]\ [\![((Z_1/Z_2)\backslash Z_3)]\!]$, pero para obtener $[\![Z_3]\!]$ y $[\![((Z_1/Z_2)\backslash Z_3)]\!]$ la definición 8.3 se utilizó menos de n veces, así por hipótesis de inducción existen árboles de prueba de los cuales son raices, a saber $\dfrac{\overset{\vdots}{Z_3} \quad ((Z_1/\overset{\vdots}{Z_2})\backslash Z_3)}{}$, ahora aplicando la regla de aplicación funcional para categorías sintácticas obtenemos el árbol $\dfrac{\overset{\vdots}{Z_3} \quad ((Z_1/\overset{\vdots}{Z_2})\backslash Z_3)}{(Z_1/Z_2)}$,

g) $D_{Z_2 \to Z_1} \overset{8.1}{=} [\![(Z_1\backslash Z_2)]\!] \overset{8.3}{=} [\![((Z_1\backslash Z_2)/Z_3)\ Z_3]\!] \overset{8.2}{=} [\![((Z_1\backslash Z_2)/Z_3)]\!]\ [\![Z_3]\!]$, pero para obtener $[\![((Z_1\backslash Z_2)/Z_3)]\!]$ y $[\![Z_3]\!]$ la definición 8.3 se utilizó menos de n veces, así por hipótesis de inducción existen árboles de prueba de los cuales son raices, a saber $\dfrac{((Z_1\backslash \overset{\vdots}{Z_2})/Z_3) \quad \overset{\vdots}{Z_3}}{}$, ahora aplicando la regla de aplicación funcional para categorías sintácticas obtenemos el árbol $\dfrac{((Z_1\backslash \overset{\vdots}{Z_2})/Z_3) \quad \overset{\vdots}{Z_3}}{(Z_1\backslash Z_2)}$,

h) $D_{Z_2 \to Z_1} \overset{8.1}{=} [\![(Z_1\backslash Z_2)]\!] \overset{8.3}{=} [\![Z_3\ ((Z_1\backslash Z_2)\backslash Z_3)]\!] \overset{8.2}{=} [\![Z_3]\!]\ [\![((Z_1\backslash Z_2)\backslash Z_3)]\!]$, pero para obtener $[\![Z_3]\!]$ y $[\![((Z_1\backslash Z_2)\backslash Z_3)]\!]$ la definición 8.3 se utilizó menos de n veces, así por hipótesis de inducción existen árboles de prueba de los cuales son raices, a saber $\dfrac{\overset{\vdots}{Z_3} \quad ((Z_1\backslash \overset{\vdots}{Z_2})\backslash Z_3)}{}$, ahora aplicando la regla de aplicación funcional para categorías sintácticas obtenemos el árbol $\dfrac{\overset{\vdots}{Z_3} \quad ((Z_1\backslash \overset{\vdots}{Z_2})\backslash Z_3)}{(Z_1\backslash Z_2)}$.

□

Con respecto a la semántica de monotonía en lenguaje natural, recordemos (ver 2.4.1) que cada tipo está parcialmente ordenado de la manera siguiente:

1. Si $X, Y \in D_e$ entonces $X \leq_e Y$ si y sólo si $X = Y$,
2. Si $X, Y \in D_t$ entonces $X \leq_t Y$ si y sólo si $X = 0$ o $Y = 1$,
3. Si $X, Y \in D_{\alpha\to\beta}$ entonces $X \leq_{\alpha\to\beta} Y$ si y sólo si para todo $x \in D_\alpha, X(x) \leq_\beta Y(x)$.

A partir de estas ordenaciones se definen los conceptos de función monótona creciente, monótona decreciente o no monótona, como a continuación.

Una función $f \in D_{\alpha\to\beta}$ es:

monótona creciente si y sólo si para todo $X, Y \in D_\alpha, X \leq_\alpha Y$ implica que $f(X) \leq_\beta f(Y)$ (notación $f \uparrow$);

monótona decreciente si y sólo si para todo $X, Y \in D_\alpha, X \leq_\alpha Y$ implica que $f(Y) \leq_\beta f(X)$ (notación $f \downarrow$);

no monótona si y sólo si no es monótona creciente ni monótona decreciente.

Una vez que se obtienen las polaridades de cada constituyente de una oración (ver 2.4.3) queremos definir cuando una oración N' es inferida a partir de una oración N, pero para lograrlo tenemos que precisar lo que vamos a entender por implicación entre dos expresiones de lenguaje natural, empecemos definiendo el concepto de subexpresión.

Definición 8.4. Sea $N = w_1 w_2 \cdots w_n, n \geq 1$, una expresión en lenguaje natural donde cada palabra $w_i \in D_{\alpha_i}$, decimos que M es una *subexpresión* de N si y sólo si se cumple una de las siguientes:

1. $M = w_i, 1 \leq i \leq n$;
2. $M = w_i M', 1 \leq i \leq n-1$, donde M' es una subexpresión de N y se cumple que ($w_i \in D_{\alpha\to\beta}$ y $M' \in D_\alpha$) o ($w_i \in D_\alpha$ y $M' \in D_{\alpha\to\beta}$);
3. $M = M' w_i, 2 \leq i \leq n$, donde M' es una subexpresión de N y se cumple que ($w_i \in D_{\alpha\to\beta}$ y $M' \in D_\alpha$) o ($w_i \in D_\alpha$ y $M' \in D_{\alpha\to\beta}$).

□

Ejemplo 8.2. Sea N = *Sue caught every armadillo*, de acuerdo al inciso 1 de la Definición 8.4 tenemos que *Sue, caught, every* y *armadillo* son subexpresiones de N; auxiliándonos de la Figura 8 vemos que *every armadillo* también es una subexpresión por el inciso 2 de la Definición 8.4, ya que *every:* $(((t/e)\backslash((t\backslash e)/e))/(t/e))$ y *armadillo:* (t/e), por el mismo inciso son también subexpresiones *caught every armadillo* y *Sue caught every armadillo*.

□

Como notación, cuando decimos que $N(M)$ es una expresión en lenguaje natural debemos entender además que tiene a M como subexpresión.

Ya que hemos precisado el concepto de subexpresión podemos a su vez precisar como podemos inferir *linealmente en un paso* una expresión de otra, la idea detrás de esta definición es conectar la noción de polaridad y la de orden parcial.

Definición 8.5. Sean $N(M)$ y $N(M')$ dos expresiones en lenguaje natural con $M, M' \in D_\alpha$, definimos que $N(M)$ *implica linealmente en un paso a* $N(M')$ (en símbolos $N(M) \overset{1}{\vDash} N(M')$) de la siguiente manera:

$$N(M) \overset{1}{\vDash} N(M') \text{ si y sólo si } \begin{cases} M \text{ tiene polaridad positiva y } M \leq_\alpha M', \text{ o} \\ M \text{ tiene polaridad negativa y } M' \leq_\alpha M. \end{cases}$$

□

Ejemplo 8.3. Recordemos que el resumen de polaridad para la expresión en lenguaje natural *An armadillo danced* (ver Figura 7) es

$$An^+(armadillo^+)^+(danced^+)^+$$

Si N = *An armadillo danced*, M = *armadillo* y M' = *mammal* entonces podemos decir que *An armadillo danced* $\overset{1}{\vDash}$ *A mammal danced* ya que *armadillo* tiene polaridad positiva y *armadillo* $\leq_{(e\to t)}$ *mammal*.

□

Ahora queremos aprovechar que las relaciones de orden parcial son transitivas para poder encadenar una serie de implicaciones. Esto lo formalizamos de la siguiente manera.

Definición 8.6. Sean $N(M)$ y $N(M')$ dos expresiones en lenguaje natural con $M, M' \in D_\alpha$, definimos que $N(M)$ *implica linealmente a* $N(M')$ (en símbolos $N(M) \overset{+}{\vDash} N(M')$) de la siguiente manera:

$$N(M) \overset{+}{\vDash} N(M') \text{ si y sólo si } \begin{cases} N(M) \overset{1}{\vDash} N(M'), \text{ o} \\ N(M) \overset{1}{\vDash} N(M'') \text{ y } N(M'') \overset{+}{\vDash} N(M'). \end{cases}$$

□

Ejemplo 8.4. Si N = *An armadillo danced*, M = *armadillo* y M' = *animal* entonces podemos decir que *An armadillo danced* $\overset{+}{\vDash}$ *An animal danced* ya que si M'' = *mammal* hemos visto en el Ejemplo 8.3 que *An armadillo danced* $\overset{1}{\vDash}$ *A mammal danced*.

También podemos decir que *A mammal danced* $\overset{+}{\vDash}$ *An animal danced* ya que a su vez *A mammal danced* $\overset{1}{\vDash}$ *An animal danced*, esto porque *mammal* tiene polaridad positiva y *mammal* $\leq_{(e\to t)}$ *animal*.

□

Para terminar, también queremos encadenar implicaciones sobre más de una subexpresión, lo cual se logra con la siguiente definición.

Definición 8.7. Sean N y N' dos expresiones en lenguaje natural, definimos que N *implica* a N' (en símbolos $N \vDash N'$) de la siguiente manera:

$$N \vDash N' \text{ si y sólo si } \begin{cases} N' = N(M') \text{ y } N(M) \overset{+}{\vDash} N(M'), \text{ o} \\ N'' = N(M''), N(M) \overset{+}{\vDash} N(M'') \text{ y } N'' \vDash N'. \end{cases}$$

□

Ejemplo 8.5. Para probar que *An armadillo danced* $\vDash$ *An animal moved* probamos primero que *An armadillo danced* $\overset{+}{\vDash}$ *An animal danced* haciendo M = *armadillo*[6] y M'' = *animal*, pero esto es justo lo que probamos en el Ejemplo 8.4; luego probamos que *An animal danced* $\vDash$ *An animal moved*, lo cual logramos al probar *An animal danced*[7] $\overset{+}{\vDash}$ *An animal moved*, que finalmente logramos al demostrar que *An animal danced* $\overset{1}{\vDash}$ *An animal moved*, que se establece porque *danced* tiene polaridad positiva y *dance* $\leq_{(e\to t)}$ *move*.

□

8.2. Arquitectura del Sistema

En la Figura 10 presentamos una arquitectura para el Sistema de Reconocimiento de Implicación Texutal con Lógica Natural.

Para obtener los tipos y el árbol de prueba de las oraciones de entrada t y h usamos las herramientas C&C[8], luego obtenemos la polaridad de cada nodo del árbol de prueba y resolvemos los elementos anafóricos aplicando la Definición 8.3.

[6] Note que empezamos con la subexpresión *armadillo*.

[7] Note que cambiamos a la subexpresión *danced*.

[8] `http://svn.ask.it.usyd.edu.au/trac/candc`

Sean $\{e_1, e_2, \ldots, e_n\}$ las subexpresiones de t que tienen polaridad positiva, a las cuales denotaremos mediante $e_i^+, i = 1, \ldots, n$; se toma la primera subexpresión, es decir, $i \leftarrow 1$, mediante WordNet[9], por ejemplo, busca una $e_{i'}$ tal que $e_i \leq e_{i'}$, si no la encuentra incrementa i y vuelve a checar, hasta que todas las subexpresiones se hayan agotado, cuando se agotan todas las subexpresiones de t que tienen polaridad positiva se anuncia que la oración t no implica a la oración h, dice el motivo y termina.

Si existe una $e_{i'}$ tal que $e_i \leq e_{i'}$, entonces en t cambiamos e_i por $e_{i'}$, como notación $t[e_{i'}/e_i]$, si $t[e_{i'}/e_i] = h$, entonces se anuncia que la oración t implica a la oración h y termina; en caso contrario $e_{i'}$ será la nueva e_i y se repite el ciclo que busca una $e_{i'}$ tal que $e_i \leq e_{i'}$.

9. Cronograma

Los años están divididos de acuerdo a los cuatrimestres oficiales del INAOE, las actividades marcadas con números refieren a los punto de la sección 7. El resto corresponden a la siguiente lista:

a. Buscar y revisar bibliografía sobre la tarea reconocimiento de implicación textual.

b. Seleccionar las propuestas para RTE que usan algún tipo de lógica.

c. Buscar y revisar bibliografía de los formalismos lógicos que se han utilizado en el tratamiento de RTE.

d. Buscar y revisar bibliografía de algunas formas de resolver el fenómeno lingüístico de Anáfora.

e. Identificar la(s) forma(s) de Anáfora que tiene(n) más presencia en los retos de RTE.

f. Buscar y revisar bibliografía sobre los tipos de Lógica Natural que hay.

g. Identificar los tipos de Lógica Natural que tienen asociada una teoría de modelos.

h. Publicar las contribuciones conforme se vayan realizando.

i. Escribir la tesis de acuerdo a los logros obtenidos.

j. Someter a revisión la tesis y atender las correcciones y sugerencias que marque el jurado.

k. Defender la tesis.

[9] `https://wordnet.princeton.edu/`

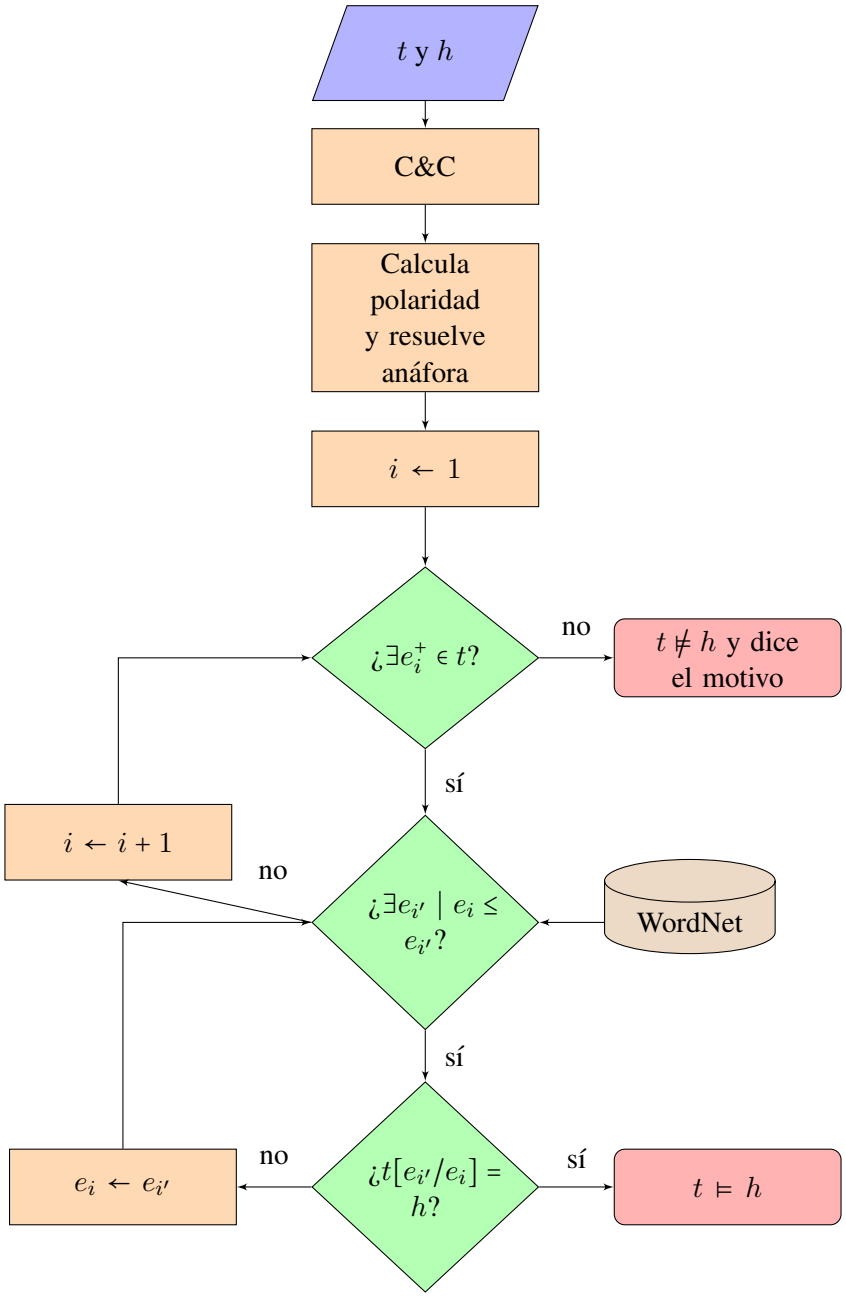

Figura 10. Arquitectura del sistema de Reconocimiento de Implicación Textual.

Año	2014			2015			2016			2017		
Actividad	I	II	III	I	II	III	I	II	III	I	II	III
a	X	X	X	X	X	X	X	X	X	X	X	
b	X	X	X									
c	X	X	X									
d	X	X	X	X	X	X						
e	X	X	X	X	X							
f	X	X	X	X								
g	X	X	X	X								
1				X	X	X						
2				X	X	X						
3				X	X	X						
4							X	X	X			
5							X	X	X			
6							X	X	X			
7								X		X		
8								X		X		
h						X			X			X
i						X			X			X
j										X	X	
k												X

Referencias

[Akh05] Elena Akhmatova. Textual entailment resolution via atomic propositions. In *Proceedings of the First PASCAL Challenges Workshop on Recognizing Textual Entailment,*, 2005.

[AM10] Ion Androutsopoulos and Prodromos Malakasiotis. A survey of paraphrasing and textual entailment methods. *Journal of Artificial Intelligence Research*, 38(1):135–187, 2010.

[BBF+05] Samuel Bayer, John Burguer, Lisa Ferro, John Henderson, and Alexander Yeh. Mitre's submissions to the eu pascal rte challenge. In *Proceedings of the First PASCAL Challenges Workshop on Recognizing Textual Entailment,*, 2005.

[BCM92] J.R. Burch, E. M. Clarke, and K. L. McMillan. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, 1992.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[BM06a] Johan Bos and Katja Markert. Recognising textual entailment with robust logical inference. In *MLCW 2005, volume LNAI 3944*, pages 404–426, 2006.

[BM06b] Johan Bos and Katja Markert. When logical inference helps determining textual entailment (and when it doesn't). In *Proceedings of the Second Challenge Workshop, Recognizing Textual Entailment*. Pascal, 2006.

[Bos13] Johan Bos. Is there a place for logic in recognizing textual entailment? *Linguistic Issues in Language Technology*, 9(3), July 2013.

[CH09] Peter Clark and Phil Harrison. An inference-based approach to recognizing entailment. In *Proceedings of the Second Text Analysis Conference (TAC 2009)*, pages 63–72, 2009.

[DDMR09] I. Dagan, B. Dolan, B. Magnini, and D. Roth. Recognizing textual entailment: Rational, evaluation and approaches. *Journal of Natural Language Engineering*, 15:459–476, 2009.

[Dek08] Paul Dekker. A guide to dynamic semantics. Technical report, ILLC/Department of Philosophy, University of Amsterdam, 2008.

[Dow94] David Dowty. The role of negative polarity and concord marking in natural language reasoning. In *Proceedings of the 4th. Conference on Semantics and Theoretical Linguistics*, Rochester, NY, 1994. Cornel University, CLC Publications.

[DRSZ13] Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. *Recognizing Textual Entailment: Models and Applications*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2013.

[dSBGP+05] Rodrigo de Salvo Braz, Roxana Girju, Vasin Punyakanok, Dan Roth, and Mark Sammons. An inference model for semantic entailment in natural language. In Joaquin Quiñonero Candela, Ido Dagan, Bernardo Magnini, and Florence d'Alché Buc, editors, *MLCW*, volume 3944 of *Lecture Notes in Computer Science*, pages 261–286. Springer, 2005.

[GB13] Swapnil Ghuge and Arindam Bhattacharya. Survey in textual entailment, 2013. Disponible en `http://www.cfilt.iitb.ac.in/resources/surveys/TE-LiteratureSurvey-2013-Swapnil_Ghuge.pdf`.

[GS91] Jeroen Groenendijk and Martin Stokhof. Dynamic predicate logic. *Linguistics and Philosophy*, 14(1):39–100, 1991.

[HCFM06] Daniel Hodges, Christine Clark, Abraham Fowler, and Dan Moldovan. Applying cogex to recognize textual entailment. In Joaquin Quiñonero-Candela, Ido Dagan, Bernardo Magnini, and Florence d'Alché Buc, editors, *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, volume 3944 of *Lecture Notes in Computer Science*, pages 427–448. Springer Berlin Heidelberg, 2006.

[JM09] Daniel Jurafsky and James H. Martin. *Speech and Language Processing, An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Pearson Education Inc., second edition, 2009.

[KP12] Alexander Koller and Manfred Pinkal. Semantic research in computational linguistics. In Claudia Maienborn, Klaus von Heusinger, and Paul Portner, editors, *Semantics: An International Handbook of Natural Language Meaning*, HSK Handbooks of Linguistics and Communication Science Series. Mouton de Gruyter, 2012.

[KVGR11] Hans Kamp, Josef Van Genabith, and Uwe Reyle. Discourse representation theory. In Dov M. Gabbay and Franz Guenthner, editors, *Handbook of Philosophical Logic*, volume 15 of *Handbook of Philosophical Logic*, pages 125–394. Springer Netherlands, 2011.

[LMMyGJS+14] José-de-Jesús Lavalle-Martínez, Manuel Montes-y Gómez, Héctor Jiménez-Salazar, Luis Villaseñor-Pineda, and David Pinto-Avendaño. Algunas semánticas lógicas para reconocer implicación textual. In David Pinto and Darnes Vilariño, editors, *Avances en la Ingeniería del Lenguaje y del Conocimiento*, volume 85, pages 33–44. Research in Computing Science, CIC-IPN, Deciembre 2014.

[Mac09] Bill MacCartney. Natural language inference. Ph. D. dissertation, Stanford University, June 2009.

[Mit02] Ruslan Mitkov. *Anaphora Resolution*. Pearson Longman, London, 2002.

[Mit03] Ruslan Mitkov. *The Oxford Handbook of Computational Linguistics (Oxford Handbooks in Linguistics S.)*. Oxford University Press, 2003.

[MM07] Bill MacCartney and Christopher D. Manning. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 193–200, Prague, June 2007. Association for Computational Linguistics.

[MR12] Richard Moot and Christian Retoré. *The Logic of Categorial Grammars, A Deductive Account of Natural Language Syntax and Semantics*. Springer, 2012.

[Mus96] Reinhard Muskens. Combining montague semantics and discourse representation. *Linguistics and Philosophy*, 19:143–186, 1996.

[RMMG08] Vasile Rus, Philip M. McCarthy, Danielle S. McNamara, and Arthur C. Graesser. A study of textual entailment. *International Journal on Artificial Intelligence Tools*, 17(4):659–685, 2008.

[RNM05] Rajat Raina, Andrew Y. Ng, and Christopher D. Manning. Robust textual inference via learning and abductive reasoning. In Manuela M. Veloso and Subbarao Kambhampati, editors, *AAAI*, pages 1099–1105. AAAI Press / The MIT Press, 2005.

[Sam11] Mark Sammons. Transformation- and logic-based approaches in rte. LSA Institute Workshop on Semantics for Textual Inference, University of Illinois at Urbana-Champaign', 2011. Disponible en `http://cogcomp.cs.illinois.edu/member_pages/sammons/files/entailment-approaches.pdf`, última visita 9 de enero de 2015.

[SB11] Mark Steedman and Jason Baldridge. Combinatory categorial grammar. In Robert Borsley and Kersti Borjars, editors, *Non-Transformational Syntax: Formal and Explicit Models of Grammar*. Wiley-Blackwell, 2011.

[vE07] Jan van Eijck. Natural logic for natural language. In Balder ten Cate and Henk Zeevat, editors, *6th International Tbilisi Symposium on Logic, Language, and Computation Batumi, Georgia*, pages 216–230. Springer, 2007.