

# Ejemplo de utilización de Markov Logic Network para resolver un problema de IoT

J.I. Estévez-Damas<sup>a</sup>

<sup>a</sup>*Universidad de La Laguna, Spain*

---

*Keywords:* Internet Of Things, Representación, Lógica de primer orden, Red lógica de Markov, Ideas preliminares

---

## 1. Introducción

Uno de los problemas típicos de la IOT consiste en lograr que los “objetos inteligentes” de la red tomen decisiones a partir de la información obtenida de diversas fuentes: conocimiento previo, evidencias cosechadas por los objetos con capacidad de sensar e interacción con el usuario. Voy a describir y comentar en esta sección un enfoque basado en herramientas propias de la IA:

- **Representación.** La lógica de primer orden será el lenguaje en el que expresaremos el conocimiento previo y las reglas de decisión. Es un lenguaje que permite reflejar de forma precisa información general sobre conjuntos de objetos, y posee esquemas de resolución a partir de los cuáles se pueden realizar inferencias.
- **Manejo de la incertidumbre y las inconsistencias.** En la IoT no podemos esperar un conocimiento preciso y plenamente consistente del mundo. Por ello es necesario emplear herramientas de la IA que faciliten el razonamiento ante esta problemática. En este ensayo proponemos el uso de la denominada “Markov Logic Network” (MLN). Esta forma de representar conocimiento integra la lógica de primer orden con la representación de la incertidumbre, usando la inferencia estadística como fundamento para la inferencia.
- **Aprendizaje.** Nos planteamos analizar un tipo de aprendizaje muy relacionado con el concepto de MLN. Se trata de aprender la “compatibilidad” de reglas predefinidas con las evidencias capturadas. Así,

24       aquellas reglas más coherentes con la evidencia jugarán un papel mayor  
25       en la inferencia y por tanto en la toma de decisiones.

26       La arquitectura sobre la que vamos a reflexionar inicialmente se compone  
27       de un bloque “Sensado del Entorno”, un bloque “Base de conocimiento”, un  
28       bloque “Inferencia” y un bloque “Actuación”. Entre el bloque de “Sensado  
29       del Entorno” y la “Base de Conocimiento” tenemos el “Interfaz de Sensado”  
30       que transforma las información de los sensores en predicados. Este interfaz,  
31       también puede utilizar información de la “Base de conocimiento” existente.  
32       Entre el bloque “Actuación” y el bloque de “Inferencia” tenemos el “Interfaz  
33       de inferencia” que puede ordenar consultas al bloque de inferencia y trans-  
34       formar predicados en acciones a aplicar por el bloque “Actuación”.

35       El modelo de funcionamiento sigue un ciclo simple: el bloque de sensado  
36       obtiene información del entorno. El interfaz de sensado realiza a continuación  
37       su labor, dando como resultado la actualización de la base de conocimiento.  
38       A continuación entra en juego el interfaz de inferencias. Este interfaz realiza  
39       un conjunto de consultas al bloque de inferencia. Estas consultas tienen  
40       como objetivo obtener información sobre los predicados del tipo “generar”.  
41       Los predicados de tipo “generar” relacionan tipos de eventos y elementos  
42       actuadores, así que básicamente son instancias concretas de acciones. Estas  
43       acciones son comunicadas al bloque de “Actuación” que las ejecuta. Esto  
44       finaliza el ciclo.

45       De momento en esta arquitectura no se incluyen bloques de aprendizaje  
46       automático, ya que en esta primera parte nos vamos a centrar en la repre-  
47       sentación y la inferencia. Posteriormente, tras describir la parte correspon-  
48       diente a la inferencia, nos ocuparemos de las posibilidades de aprendizaje  
49       automático.

## 50   **2. Utilizando la lógica de primer orden**

51       Planteemos el escenario. Tenemos una casa, donde se usan etiquetas  
52       electrónicas para generar eventos, hay sensores del ambiente y actuadores,  
53       así como elementos con capacidad de procesamiento capaces de acumular la  
54       información pertinente para la toma de decisiones (evidencias sobre el estado  
55       físico del entorno, base de conocimiento previo). Vamos a crear un modelo  
56       de descripción, basado en lógica de primer orden con tipos.

57       Consideremos los siguientes tipos:

$$\mathcal{T} = \{lugar, sensor, actuador, evento, magnitud, valor, clase\}$$

58 El conjunto de constantes  $\mathcal{C}$  puede ser particionado utilizando los tipos  
59 en  $\mathcal{T}$ .

60 Además tendremos un conjunto de predicados:

- 61 • **localización(sensor/actuador,lugar)**. Establece una relación espa-  
62 cial entre un sensor o actuador y un lugar.
- 63 • **eventoactivo(evento,sensor)**. Señala un evento detectado en el sis-  
64 tema que debe ser procesado
- 65 • **despuésde(evento,evento)**. Establece una relación temporal entre  
66 dos eventos.
- 67 • **generar(evento,actuador)**. Esta relación implica la producción in-  
68 mediata de un evento por parte de un actuador.
- 69 • **medida(sensor,magnitud,valor)**. Establece la medición de un valor  
70 para una magnitud por parte de un sensor.
- 71 • **tiposensor(sensor,clase)**. Permite asignar una o varias clases a un  
72 objeto.
- 73 • **admiteactuacion(actuador,clase)**. Permite asignar una o varias  
74 clases a un actuador.

75 Supongamos ahora que queremos diseñar una regla que permita encen-  
76 der la iluminación de una habitación cuando se produzca la activación del  
77 interruptor basado en etiqueta RFID de la habitación y no exista suficiente  
78 claridad. Podríamos tener algo así:

$$\begin{aligned} &localizacion(S, H) \wedge tiposensor(S, rfidbasedswitch) \wedge \\ &eventoactivo(defaultevent, S) \wedge localizacion(S1, H) \wedge \\ &medida(S1, lightmeasurement, lowvalue) \wedge localizacion(A, H) \wedge \\ &admiteactuacion(A, roomlightsactuator) \\ &\Rightarrow generar(ev_turnonroomlights, A) \end{aligned}$$

79 Bajo la interpretación clásica de la lógica de primer orden, esta regla  
80 viene a indicar que cualquier “mundo” donde los predicados grounded del

81 antecedente de la regla sean verdaderos y el consecuente falso es imposible.

82 Por ejemplo, no puede suceder:

$localizacion(interruptor13, cocina) = T$   
 $tiposensor(interruptor13, rfidbasedswitch) = T$   
 $eventoactivo(defaultevent, interruptor13) = T$   
 $localizacion(senselight1, cocina) = T$   
 $medida(senselightcocina, lightmeasurement, lowvalue) = T$   
 $localizacion(smartbulb1, cocina) = T$   
 $admiteactuacion(smartbulb, roomlightsactuator) = T$

83 y al mismo tiempo:

$generar(ev_turnonroomlights, smartbulb) = F$

84 Si los predicados del antecedente efectivamente se prueba que son ver-  
85 daderos (bien, porque sean evidencias o porque se han deducido), entonces,  
86 en todos los “mundos” posibles debe ocurrir que:

$generar(ev_turnonroomlights, smartbulb) = T$

87 Un método de inferencia válido (soundness) y completo (completeness)  
88 debe ser capaz de obtener esta conclusión.

89 Un aspecto a considerar se refiere al papel de las constantes en esta clase  
90 de representaciones. El análisis de una regla, teniendo en cuenta los objetivos  
91 que queremos cumplir puede llevarnos observar que la regla:

92 • **Puede dar lugar a comportamiento indeseados.** La general-  
93 ización de la regla podría ser excesiva. Por ejemplo, supongamos que  
94 en una habitación hay dos interruptores de tipo *rfidbasedswitch*, uno  
95 destinado a conectar la climatización y otro destinado a encender las  
96 luces. Según la regla descrita, si el usuario lanza el evento usando  
97 el rfid para climatización se inferiría el predicado asociado a la gen-  
98 eración del evento “encender luces”. Este tipo de problemas, se puede  
99 resolver creando tipologías más precisas que describan roles particu-  
100 lares. Introducir esta información en el sistema puede ser cosotoso y  
101 no estar exenta de errores, especialmente ante reconfiguraciones del sis-  
102 tema. En el ejemplo que estamos describiendo, el diseñador del sistema  
103 asume que todo interruptor rfid está destinado a encender luces en las

104 habitaciones. El problema se resuelve incluyendo un nuevo tipo “más  
105 específico”:

$$rfidswitch\_turnonlight$$

106 A la hora de facilitar el diseño de los roles, lo conveniente sería crear  
107 una jerarquía de roles de modo que si por ejemplo, el usuario introduce  
108 en la base de conocimiento:

$$tiposensor(interruptor13, rfidswitch\_turnonlight)$$

109 automáticamente se incluyese la evidencia:

$$tiposensor(interrutpor13, rfidbasedswitch)$$

110 Esto se puede conseguir mediante inferencia lógica si se incorpora a la  
111 base de conocimiento una regla como:

$$tiposensor(X, rfidswitch\_turnonlight) \Rightarrow tiposensor(X, rfidbasedswitch)$$

112 • **Puede no ser suficientemente descriptiva.** Supongamos ahora que  
113 existe una habitación donde hay varios actuadores capaces de encender  
114 una luz. Según la regla que hemos escrito, todos estos actuadores  
115 producirían su evento asociado. Supongamos que sólo queremos que  
116 un tipo concreto de actuador sea el que realice la operación. En este  
117 caso se podrían crear tipos de actuación más particulares, por ejemplo  
118 podríamos tener en los antecedentes:

$$admiteactuacion(A, roomlightsceiling)$$

119 Iguamente se pueden incluir predicados para categorizar las habita-  
120 ciones.

121 Veamos ahora algunas características de la representación en lógica de  
122 primer orden con el ejemplo presentado:

- 123 • **Se trata de una representación compacta.** Esto se debe al uso  
 124 de las variables lógicas. Aunque no se representa explícitamente, la  
 125 fórmula anterior está encabezada por  $\forall S, H, S1, A$ , por lo que sería  
 126 aplicable a todas las habitaciones, sensores y actuadores que cumplan  
 127 la relaciones cuya estructura se define en la regla: todo sensor rfid que  
 128 compata habitación con un actuador catalogado como capaz de dar luz  
 129 en una habitación y tenga un evento activo va a permitir inferir la gen-  
 130 eración de un evento de encendido en el actuador. Si este tipo de regla  
 131 fuera representada usando lógica proposicional habría que haber creado  
 132 reglas específicas para cada caso, o bien introducir nuevos predicados.
- 133 • **Consistencia.** La base de reglas debe ser consistente. Es decir, si  
 134 hay reglas contradictorias no encontraremos ningún “mundo” posible  
 135 y no deducirá ningún predicado. Esto complica el diseño de las reglas,  
 136 ya que en términos “humanos” las especificaciones pueden incluir esta  
 137 clase de inconsistencias. Esto puede ser ventajosos en algunos ámbitos,  
 138 pero no en otros. Por ejemplo, supongamos la siguiente regla:

$$\begin{aligned} &tiposensor(S, smartclock) \wedge eventoactivo(nightdetected, S) \wedge \\ &localizacion(S1, H) \wedge tipohabitacion(H, daytimerooms) \wedge \\ &tiposensor(S1, movementsensor) \wedge medida(S1, vibration, zerovalue) \\ &\Rightarrow generar(ev\_turnoffroomlights, A) \end{aligned}$$

139 Y además tenemos una regla que impide generar eventos mutuamente  
 140 excluyentes como:

$$generar(ev\_turnonlights, A) \Leftrightarrow \neg generar(ev\_turnofflights, A)$$

141 Como vemos, en ese caso podríamos llegar a una contradicción. Evitar  
 142 estas contradicciones es posible incluyendo más reglas y predicados,  
 143 pero el precio a pagar es el aumento en la complejidad de la base de  
 144 reglas.

- 145 • **Inclusión en un agente.** Una cuestión a resolver es cómo exacta-  
 146 mente incluir esta representación en un agente. Podemos distinguir  
 147 entre dos aproximaciones. Por una parte, tenemos estrategias en las  
 148 que el agente mantiene una imagen del estado a partir de dos fuentes:

149 externamente el conjunto de evidencias se actualiza e internamente in-  
 150 fiere hechos sobre el estado a partir de las evidencias y de su base de  
 151 reglas (al menos en una parte). Por otra parte, tenemos aquellas en las  
 152 que el agente no infiere los cambios en el estado y se limita a actualizar  
 153 su base de evidencias desde una fuente externa tras el accionamiento  
 154 de un actuador.

155 A la primera aproximación corresponden formalismos como el cálculo  
 156 situacional o el cálculo de fuentes, esquemas ampliamente usados en  
 157 robótica, ambos englobados en lo que se conoce métodos de razon-  
 158 amiento sobre acciones. El cálculo situacional tiene como principales  
 159 ventajas, la inferencia de características del mundo sin necesidad de  
 160 fuentes externas que lo establezcan. Esto le permite incorporar más  
 161 información con diferentes grados de abstracción, lo que permite ra-  
 162 zonamientos más complejos. Por ejemplo, en el mundo de los bloques  
 163 podríamos tener un sensor para determinar exactamente la posición de  
 164 cada bloque. El sistema podría inferir internamente la altura de las  
 165 torres de bloques y cómo esa altura cambia al reposicionar los bloques.  
 166 El concepto de altura de la torre, es una abstracción que permite definir  
 167 mecanismos de decisión más compactos para determinados problemas.  
 168 Otra ventaja es la posibilidad de detectar fallos en el funcionamiento  
 169 del agente. Supongamos la regla:

$$\begin{aligned} &ocalizacion(S, H) \wedge tiposensor(S, lightsensor) \wedge \\ &localizacion(A, H) \wedge admiteactuacion(A, roomlightsactuator) \wedge \\ &turnedon(A) \wedge medida(S, lightmeasurement, lowvalue) \\ &\Rightarrow faultdetection(A, S, H) \end{aligned}$$

170 Esta regla va a determinar que en cualquier habitación donde haya un  
 171 actuador del tipo *roomlightsactuator*, que se ha activado, si hay un  
 172 sensor de luz detectando un valor bajo, se infiere una detección de un  
 173 posible fallo. La clave para lograr esta detección es la actualización del  
 174 estado después de haber activado el actuador, es decir, debe aparecer  
 175 el predicado *turnedon* con el actuador correspondiente en la base de  
 176 conocimiento. El cálculo situacional establece un marco para razonar  
 177 con acciones que alteran el estado, permitiendo incluir y eliminar pred-  
 178 icados de la base de conocimiento.

179 En este documento, sin embargo, nos vamos a centrar en la segunda de  
 180 las aproximaciones. Vamos a suponer, que el agente sólo modifica su

base de conocimiento mediante una fuente externa. Esto simplificará el conjunto de reglas permitièndonos centrarnos en las posibilidades del resto de tecnologías que queremos explorar. En la arquitectura descrita en la introducción, se correspondería con el “Interfaz de sensado”.

### 3. Representación mediante una Markov Logic Network

La base de reglas puede verse como la definición de un conjunto de restricciones que limitan los “mundos” posibles (asignaciones de verdad a los predicados “grounded”, donde las variables son sustituidas por constantes). Así, hay asignaciones para los que el conjunto de fórmulas que componen la base de conocimiento son verdaderas y otras en las que no. Las asignaciones que mantienen el valor “True” en todas las fórmulas de la base de conocimiento constituyen los “mundos” posibles.

Las redes lógicas de Markov o MLN introducen otra filosofía diferente. Una MLN también contiene un conjunto de fórmulas en lógica de primer orden, y también establece un grado de verdad para los diferentes mundos. La diferencia es que el grado de verdad asignado no es “T” o “F”, sino un valor de **probabilidad**. Así, simplemente habrá mundos más probables que otros. A la hora de consultar si la base de conocimiento puede derivar una fórmula, lo que se obtiene es una probabilidad para la fórmula. En realidad, la definición de la MLN es el conjunto de fórmulas con un peso asignado a cada uno de ellas. El significado de este peso, es que mientras mayor sea, la fórmula asociada tendrá que cumplirse para aquellos mundos que tengan una mínima probabilidad. En el caso límite, si todos los pesos tienden a infinito volvemos a la lógica clásica de primer orden.

La asignación de probabilidad a los mundos, es decir, la definición de una distribución de probabilidad para la asignación de verdad al mundo, se basa en un modelo matemático usado en otros campos (por ejemplo, física), denominado red de Markov (MN). La red de Markov es un grafo no dirigido formado por nodos y arcos que únen estos nodos. Se trata de un modelo gráfico de probabilidad, junto con un conjunto de pesos define una distribución de probabilidad del estado conjunto de los nodos de la red. Esta distribución está factorizada y cada factor considera sólo el estado de los nodos incluidos en uno de los cliques de la MN (recordemos que un clique en un grafo es un subconjunto de los nodos, tales que existe un arco entre cualquier par).

Así, una red lógica de Markov, con su conjunto de fórmulas pesadas y las constantes que pueden sustituir a las variables lógicas, puede verse



217 como un generador de una red de Markov donde cada nodo de esta última  
 218 es un posible predicado grounded. Los cliques se forman a partir de las  
 219 fórmulas, ya que cada fórmula genera un conjunto de fórmulas grounded y  
 220 cada una de éstas establece la interconexión por arcos de todos los pares  
 221 de predicados grounded que la forman. Sin profundizar más, ya podemos  
 222 intuir que el tamaño de la red de Markov subyacente crece con el número  
 223 de predicados grounded haciendo más costoso computacionalmente cualquier  
 224 cálculo de inferencia probabilística.

225 Veamos algunas características de esta representación:

- 226 • **Las fórmulas que se incluyen en la base de conocimiento, no**  
 227 **tienen por qué ser siempre ciertas.** El ejemplo estándar que se  
 228 suele poner es el siguiente:

$$\begin{aligned} 0.8 \text{ friends}(X, Y) \wedge \text{smokes}(X) &\Rightarrow \text{smokes}(Y) \\ 0.3 \text{ smokes}(X) &\Rightarrow \text{cancer}(X) \end{aligned}$$

229 La primera regla nos dice que para cualquier par de amigos, si uno  
 230 fuma el otro también. El peso asociado es 0.8. Si tenemos en nuestra  
 231 base de evidencias, los predicados *friends(Bob, Ann)* y *smokes(Bob)*,  
 232 al consultar *smokes(Ann)* lo que se obtiene es una probabilidad, que  
 233 será más alta o más baja en función de la importancia relativa del peso  
 234 asociada a la regla, y del resto de reglas incluídas. La segunda regla  
 235 nos dice que todo el que fuma tiene cancer, pero luego le asigna un peso  
 236 más bajo. Así, aunque quizás *smokes(Bob)*, tenga una probabilidad  
 237 relativamente alta, la inferencia *cancer(Bob)* nos dé una probabilidad  
 238 que puede ser relativamente baja.

239 En este ejemplo, vemos que es posible representar información similar a  
 240 las “creencias” que no son siempre ciertas pero que nuestra experiencia  
 241 les otorga cierto grado de verosimilitud.

- 242 • **La base de reglas puede incluir fórmulas contradictorias entre**  
 243 **si.** Será la importancia relativa de los pesos la que determine cuál de las  
 244 fórmulas será más relevante. Esto elimina, la necesidad de un modelado  
 245 “detallado” y cuidadoso requerido en las representaciones basadas en  
 246 lógica de primer orden para evitar contradicciones.

- 247 • **Procedimientos de inferencia.** Hay que tener en cuenta que los pro-  
 248 cedimientos de inferencia, requieren búsquedas sobre el espacio de los

249 mundos. Este espacio de búsqueda puede ser enorme, ya que aumenta  
250 exponencialmente con el número de posibles predicados “grounded”.  
251 Sin embargo, existen diferentes procedimientos de inferencia que pueden  
252 trabajar con base de reglas de tamaño medio de forma eficiente:

- 253 – Es razonable por ejemplo obtener el “mundo” más probable aten-  
254 diendo a la información introducida por la MLN (o dicho de otra  
255 manera, la asignación de los diferentes predicados “grounded”  
256 bajo la MLN definida). Se trata de una estimación denominada  
257 MAP (maximum a posteriori en términos de estadística bayesiana).  
258 Este procedimiento podría servir, por ejemplo, para obtener si un  
259 predicado “grounded” es verdadero o falso en el “mundo” más  
260 probable. En el modelo que hemos venido utilizando en este doc-  
261 umento, bastaría hacer una consulta sobre todos los predicados  
262 “grounded” derivados de *generar*( $E, A$ ). Estos predicados serían  
263 el conjunto de acciones recomendadas del sistema. Dicho sistema  
264 servirá para determinar cuál/es serían las siguientes acciones a  
265 aplicar por el módulo de actuación.
- 266 – También es posible obtener de forma aproximada probabilidades  
267 marginales para fórmulas, incluso probabilidades con condicionales.  
268 Para ello se usan técnicas de muestreo (métodos de Montecarlo).  
269 Así, para obtener la probabilidad de una fórmula se realiza una  
270 estimación sobre una muestra de mundos de la fracción de mun-  
271 dos donde dicha fórmula se cumple respecto al total. El muestreo  
272 lógicamente debe realizarse sobre la distribución de probabilidad  
273 de los mundos, por lo que los mundos más probables aparecerán  
274 en la muestra con mayor “facilidad”.

#### 275 4. Aprendizaje automático

276 Hasta ahora hemos esbozado un modelo que puede decidir la aplicación  
277 de determinadas acciones tomando como base la información sensada del  
278 entorno y una red lógica de Markov. Sin embargo, existen dos aspectos que  
279 no hemos tratado:

- 280 • ¿Cómo se determinan los valores de los pesos asociados a las fórmulas  
281 de una MLN?.
- 282 • ¿Qué fórmulas hay que incluir en la MLN?

283       La utilización de conocimiento previo y un ajuste basado en prueba - error  
284 podría ser una posible respuesta a las preguntas anteriores. Una alternativa  
285 a esto es el denominado aprendizaje automático. El aprendizaje automático  
286 en su modalidad supervisada requiere de:

- 287       1. Un modelo a ajustar.
- 288       2. Un conjunto de entrenamiento.
- 289       3. Un algoritmo de aprendizaje del modelo, dado el conjunto de entre-  
290       namiento.

291       Las preguntas anteriores se refieren justamente a aspectos del modelo.  
292 Queremos calcular los pesos o incluir determinadas fórmulas y dejar fuera  
293 otras para lograr que lo que del modelo se derive sea lo más cercano posible  
294 a lo obtenido en el conjunto de entrenamiento.

295       La siguiente cuestión sería: ¿quién es nuestro conjunto de entrenamiento?.  
296 Para pensar sobre esto, hemos de recordar que nuestra MLN trata de repre-  
297 sentar la distribución de probabilidad de todos los mundos posibles: habrá  
298 mundos más probables que otros y esto está determinado por la MLN.