IRASubcat
Un sistema para adquisición automática de marcos de subcategorización a partir de corpus

Autor: Ivana Romina Altamirano
Supervisora: Laura Alonso i Alemany

29 de diciembre de 2009

Agradecimientos

Agradezco especialmente a Laura Alonso i Alemany, por permitirme hacer en la tesis algo que fusionaba muchas de mis pasiones, por su paciencia, y por su buena honda!

A Gabriel Infante López, que más de una vez, cuidó a Nunu para que Laura me guíe en la tesis.

A Nunu, por portarse bien, cada vez que yo necesitaba ayuda de Laura.

Al hermano de Laura, el experto en latex.

A Toni Oliver por brindar información imprescindible para mi tesis.

A mi profe de ruso, Adriana Machado, por despertar y mantener mi interés en el idioma ruso.

A mis padres, a mi novio, y a mis amigos.

Ivana Romina Altamirano

Índice general

Capítulo 1

Introducción

1.1. Presentación del Problema

1.1.1. Algunos conceptos preliminares

Antes de explicar qué es la "Adquisición automática de marcos de Subcategorización a partir de corpus", vamos a introducir algunos conceptos que se usarán a lo largo de la tesis.

Un **corpus lingüístico** es un conjunto, normalmente muy amplio, de ejemplos reales de uso de una lengua. Estos ejemplos pueden ser textos (típicamente), o muestras orales (normalmente transcriptas). En nuestro caso van a ser oraciones en una lengua particular.

Las **oraciones** son grupos de palabras que, además de estar reunidas con sentido, transmiten un mensaje completo.

Los **verbos** son palabras que sirven para nombrar las distintas acciones que realizan las personas, los animales, las cosas. Es decir, cualquier acción, cualquier actividad que se pueda nombrar, se nombra utilizando un verbo. El verbo también puede denotar un estado, por ejemplo, el verbo "parecer" es un verbo que no denota una acción, denota un estado.

El verbo juega un rol principal en la oración, ya que cumple la función de ser la palabra más importante del predicado, es decir, su núcleo. En función de este núcleo es que se organizan las demás palabras que componen la oración, si es que las hay. Decimos esto porque puede haber predicados compuestos por una sola palabra, como, por ejemplo, en la oración: "Veremos". En estos casos, la única palabra del predicado o de la oración será un verbo.

Un **patrón** son las palabras, o algunas de las características de estas palabras, que encontramos acompañando un verbo, es algo que ocurre en una oración particular.
Veamos un ejemplo:

```
"El nene juega a la pelota"
```

En esta oración el patrón de palabras encontrado para el verbo "juega" es [El nene a la pelota].

El patrón puede encontrarse a diferentes niveles de abstracción. En el caso anterior el patrón es el de la forma de las palabras, veamos ahora otros tipos de patrones.

Para obtener diferentes niveles de abstracción vamos a etiquetar las palabras de una oración con un analizador morfosintáctico.

Un **analizador morfosintáctico** es un programa que se encarga de dar información más o menos completa sobre la forma y, dependiendo de las lenguas, también sobre la función sintáctica de una oración en un texto.

Veamos el resultado de analizar morfosintácticamente la oración anterior con el analizador TreeTagger (Schmid 1994):

```
"El  nene  juega  a   la  pelota"
  ART  NC   VLfin PREP ART NC
  el  nene  jugar  a   el  pelota
```

La segunda fila es la categoría morfosintáctica de las palabras, y la tercera fila, el lema de las palabras.

La **categoría morfosintáctica** de las palabras indica cómo se manifiestan en la forma de la palabra las diferentes categorías gramaticales que puede tomar (de ahí el aspecto morfológico) y su rol en el ordenamiento de la oración (de ahí su aspecto sintáctico).

El **lema** de la palabra es su forma canónica, en este caso, el lema de la palabra 'la' es 'el', la cual está generalmente en masculino singular en caso de ser sustantivos, y en infinitivo en caso de ser verbo.

Los patrones morfosintáctico y de lema respectivamente de la oración anterior para el verbo "juega" son los siguientes:

```
[ART NC PREP ART NC]
[el nene a el pelota]
```

Pero ahora analicemos la oración "Los nenes juegan a la pelota", veamos el resultado del análisis morfosintáctico realizado por el TreeTagger (Schmid 1994):

```
Los  nenes  juegan  a   la   pelota
ART  NC     VLfin  PREP ART  NC
el   nene   jugar   a   el   pelota
```

y así vemos como el verbo "juega" y "juegan" comparten el mismo patrón morfosintáctico y de lema de las palabras, por ello podríamos decir que el verbo "jugar" tiene en esas 2 oraciones el mismo patrón morfosintáctico [ART NC PREP ART NC], pero no el mismo patrón de forma de palabras. Vemos en este ejemplo cómo el análisis lingüístico nos proporciona una abstracción útil y que se corresponde con nuestras intuiciones sobre la lengua.

Los niveles de información que son relevantes para un lexicón verbal conciernen todos los aspectos léxicos desde morfológicos a sintácticos y semánticos.

Un **marco de subcategorización verbal** para un verbo particular es un conjunto de patrones con los que el verbo puede ocurrir en oraciones bien formadas gramaticalmente.

La información de cuáles son los patrones lingüísticos asociados a un verbo puede ser generada manualmente, pero eso es muy costoso, requiere mucho tiempo, raramente la información estará completa, y no contará con información de frecuencias, y además será dificil de actualizar.

Los problemas mencionados de la adquisición manual, la gran disponibilidad de corpus en internet y la creciente velocidad de procesamiento de las computadoras incentivan a la adquisición automática de dichos recursos, utilizando corpus existentes y algoritmos computacionales.

La información adquirida es almacenada en lexicones legibles por la máquina y es actualizada rápida y fácilmente. Además, con esta última forma de adquirir los marcos de subcategorización

verbal, también se obtiene información de las frecuencias en que los patrones coocurren con los verbos.

1.1.2. Interés de la Subcategorización en PLN

La información de subcategorización es importante en tareas computacionales y en aplicaciones de Procesamiento de Lenguaje Natural, tales como analizadores sintácticos, traducción automática y respuesta a preguntas.

Veamos un ejemplo de la utilidad de contar con información de marcos de subcategorización para un analizador sintáctico, pero antes, vamos a dar una breve explicación de que es un analizador sintáctico.

Un analizador sintáctico es un programa que se encarga de dar información más o menos completa sobre la sintaxis de una frase en un texto, es decir, describe la relación entre sus palabras. Esas tareas suponen una formalización del texto, que esta regido por una gramática formal. La estructura revelada por el análisis dá precisamente como las reglas gramaticales están instanciadas en el texto.

Para realizar el análisis de un texto, se evalúan muchas posibilidades de analizarlo, a fin de encontrar la más adecuada al texto. Pero si el analizador contara con información de marcos de subcategorización para los verbos, el número de posibles estructuras para la oración se reduciría porque se evaluarían solo las posibilidades que aparecen en el marco de subcategorización del verbo.

Otro ejemplo de aplicación es la posibilidad de identificar oraciones que no son gramaticales en un idioma (esto sería muy útil en respuesta a preguntas, si el sistema genera las respuestas mediantes técnicas de PLN). Mostramos aquí la utilidad de poseer información de marcos de subcategorización para identificar oraciones gramaticales, y oraciones que no son gramaticales.

Veamos el siguiente ejemplo:

```
Liliana compra frutas.
*Liliana camina frutas.
```

El análisis morfosintáctico dado por el TreeTagger (Schmid 1994) para la primera oración "Liliana compra frutas" fue:

```
'Liliana' NP <unknown>
'compra' VLfin comprar
'frutas' NC fruta
```

Siendo la segunda columna la categoría morfosintáctica de la palabra en la oración y la tercera el lema de la palabra.

Si tuvieramos información de marcos de subcategorización para el verbo 'comprar' veríamos que la oración anterior es correcta ya que [NP, NC] es un patrón posible para el verbo 'comprar'.

En cambio, en la segunda oración "Liliana camina frutas", el resultado del TreeTagger es el mismo para las etiquetas morfosintácticas:

```
'Liliana' NP <unknown>
'camina' VLfin caminar
'frutas' NC fruta
```

pero el marco de subcategorización del verbo 'caminar' no puede contener al patrón [NP, NC], por lo tanto deducimos que la oración no es gramatical.

Para mejorar la performance de la computadora al armar, identificar y procesar oraciones gramaticales es importante contar con información de subcategorización verbal.

1.1.3. Dos partes del problema: identificar marcos y asociar marcos a verbos

Para identificar marcos de subcategorización verbal, es necesario identificar el verbo en la oración, y luego las palabras y/o características que lo acompañan serán los componentes del patrón.

Luego de analizar muchas oraciones veremos como cada verbo tiende a repetir los mismos patrones, en distintos niveles de abstracción o focalizándose en distintas características.

Cuando la adquisición se realiza automáticamente es necesario filtrar los patrones que se asocian a los verbos, es decir, descartar posibles errores o de simples ocurrencias sin significancia estadística.

Para filtrar patrones no significativos se pueden usar distintos test, entre ellos los de frecuencias y los de hipótesis. Por ejemplo (Brent 1993) usó el test binomial, pero consideró que obtenía mejores resultados con el test de frecuencias.

En nuestro sistema usaremos test de frecuencia absoluta de verbos para descartar verbos con pocas ocurrencias en el corpus, test de frecuencia relativa de la co-ocurrencia verbo-patrón, para descartar co-ocurrencias de verbos con patrones con pocos ejemplos en el corpus y test de hipótesis likelihood ratio, por ser este test apropiado para fenómenos con pocas ocurrencias, lo que es muy común en lenguaje natural.

1.2. Descripción de la Solución

Queremos crear una herramienta que permita adquirir información de marcos de subcategorización verbal automáticamente, que funcione para cualquier idioma, que reciba corpus con distintos niveles de etiquetado, y que sea altamente parametrizable.

Para que el sistema sea altamente parametrizable, se definió un archivo de configuración para permitirle al usuario las siguientes opciones de procesamiento, las cuales explicaremos con más detalle en el capítulo 4.

- Considerar sólo un conjunto de verbos.
- Actualizar un diccionario existente
- Longitud del patrón que se considera, puede ser un número, que es la cantidad de palabras y/o características a cada lado del verbo dentro de la oración, o considerar todas las palabras de la oración.
- Si desea completar el patrón con alguna palabra comodín, en los casos que la cantidad de palabras a cada lado del verbo sea menor a la seleccionada en la opción anterior.
- Si desea tener en cuenta el orden de las palabras y/o características encontradas alrededor del verbo, o no tenerlo en cuenta. Esta opción con valor 'NO' considera como iguales a los patrones que tienen los mismos constituyentes pero en distinto orden.
- Características de las palabras que se considerarán (por ejemplo, categoría morfosintáctica, lema, etc.).

- Si se tendrán en cuenta o no la forma literal de las palabras.
- Se se introduce o no el caracter '|' para representar la posición del verbo en el patrón.
- La frecuencia absoluta mínima necesaria para considerar un verbo.
- La frecuencia relativa mínima necesaria para considerar una coocurrencia entre un verbo y un patrón.
- Si se usará el test de hipótesis Likelihood Ratio.
- Si se colapsarán patrones.

Para que el sistema pueda tratar corpus con muy distintos tipos de anotación y diferentes unidades lingüísticas, definimos un formato para el corpus de entrada al sistema, que básicamente refieren a la forma en la que se debe codificar los datos del corpus para que el sistema lo pueda procesar.

El formato del corpus que recibe el sistema es un archivo XML codificado en UTF-8 con la siguiente estructura, presentada gráficamente en la Figura 3.1:

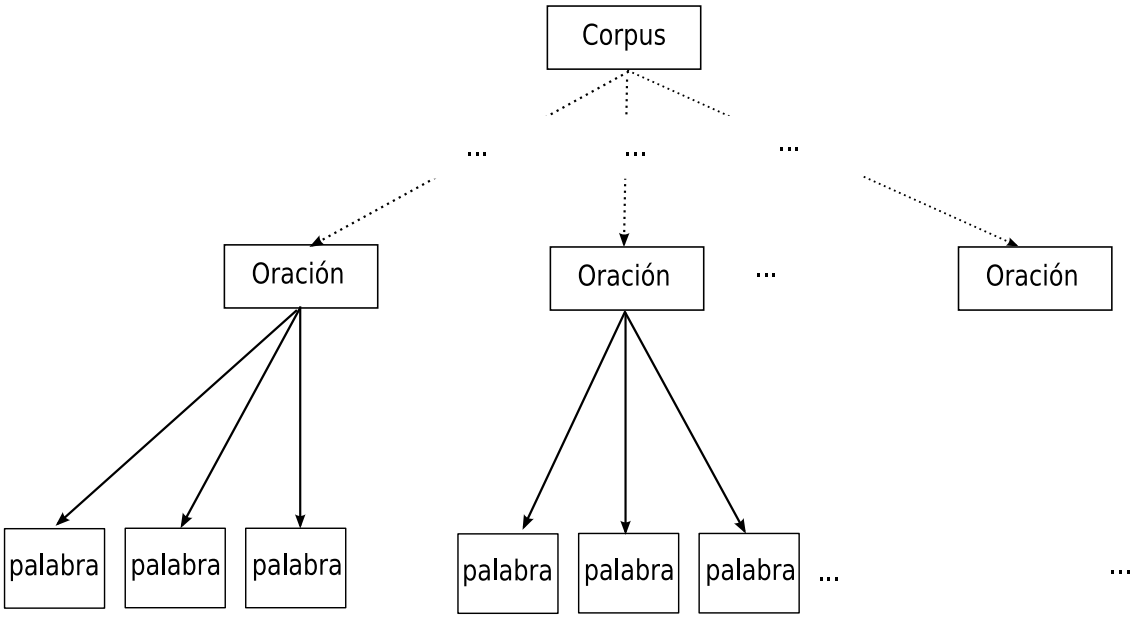

Figura 1.1: Arbol XML del Corpus

```
<corpus>
        <oracion ID='1'>
                <palabra lema='el' sint='ART'>La</palabra>
                <palabra lema='casa' sint='NC'>casa</palabra>
                ...
        </oracion ID='2'...>
        <oracion>
                <palabra ...> ... </palabra>
                <palabra ...> ... </palabra>
                ...
        </oracion ID='3'>
```

```
        .
        .
        .
        <oracion ID='1230'>
                <palabra ...> ... </palabra>
                <palabra ...> ... </palabra>
                ...
        </oracion>
        ...
</corpus>
```

Cabe destacar que los nombres 'corpus', 'oracion' y 'palabra' son puramente ilustrativos, tampoco son obligatorios los atributos que se muestran para los elementos XML.

Como mostraremos más adelante, en el capítulo 4, el sistema sólo pide que se le indique cuál es el nombre de la etiqueta padre de las etiquetas que representan los constituyentes oracionales que van a constituir el patrón verbal. El sistema considerará como elementos para formar el patrón todos los hijos de la etiqueta dada por el usuario. Los puntos suspensivos entre Corpus y Oración (en la Figura 3.1) indican que la unidad lingüística en la cual buscaremos los patrones puede estar en cualquier nivel del árbol, pero es necesario que sus hijos contengan información de los componentes de las características para los cuales se ejecuta el sistema.

El atributo ID en la etiqueta oracion, no es un atributo obligatorio, pero será necesario si el usuario quiere que el sistema le dé un archivo de salida que contiene los ID's de las oraciones en las cuales encontró cada patrón.

El corpus puede ser plano, es decir, no tener ningún tipo de análisis lingüístico asociado. En ese caso, será necesario que se marquen de alguna forma los verbos en el corpus (manualmente, con heurísticas *ad hoc*), ya que el sistema no posee heurísticas de identificación de verbos para ningún idioma. El sistema se enfoca en la tarea de encontrar información de marcos de subcategorización verbal, por lo tanto consideramos que la tarea de identificar verbos forma parte del pre-procesamiento del corpus y queda fuera del alcance de esta tesis.

El sistema está preparado para permitir corpus ricamente etiquetados, la información que puede tener el corpus puede ser de todos los tipos que el usuario considere relevante: morfosintáctica, semántica, pragmática, etc. La información asociada a los constituyentes de los patrones se expresa mediante atributos de los elementos XML que representan a esos constituyentes, como hemos visto en el ejemplo anterior con "sint" y "lema".

A partir de la información que provea el corpus y las opciones de procesamiento seleccionadas por el usuario o las provistas por defecto por el sistema, se buscarán marcos de subcategorización. Los patrones que formarán estos marcos tendrán la información especificada por el usuario, ya sea el contenido de los constituyentes, el contenido de alguno de sus atributos o combinaciones de contenido de atributos.

Para ilustrar este funcionamiento, mostramos partes de los 2 corpus a los cuales se aplicó el sistema, como se detallará en la segunda parte de la tesis. El primer corpus es un corpus plano en ruso, el cual se etiquetó con el analizador morfosintáctico automático TreeTagger. A partir del corpus analizado se construyó el archivo en XML con el siguiente formato:

```
<corpusRuso>
   ...
   <FRASE ID='156'>
      <palabra lema='он' morfosint='P-3msn'>он</palabra>
```

```
      <palabra lema='купить' morfosint='Vmis-sma-e'>купил</palabra>
      <palabra lema='дом' morfosint='Ncmsan'>дом</palabra>
      ...
   </FRASE>
   ...
</corpusRuso>
```

En este caso, la información que podremos usar para la obtención de marcos de subcategorización serán: la forma de las palabras, los valores de los atributos 'lema' y 'morfosint', y combinaciones de ellos.

El segundo corpus que se usó fue un corpus en español asociado a un rico análisis a diferentes niveles lingüísticos. Este análisis fue hecho de forma manual para cada una de las oraciones del corpus. Para este corpus tenemos el siguiente formato:

```
<corpus>
   ...
   <s ID='5453' semor1='Evento' anotado='1-17' verbo='17' lema_verbo='asegurar'
      sentido='asegurar_2' metaf='0' sujeto_elidido='1' WN_S='00598975'>
      <phr id='1' rs='T-desp' cat='OEstdir' fs='Obj Directo' Argumento='1'>
      </phr>
      <phr id='2' cat='verbo' lema_verbo='asegurar'></phr>
   </s>
   ...
</corpus>
```

La unidad lingüística a estudiar serán las 's' por lo tanto la unidad lingüística de la que se extraerán los constituyentes de los patrones serán las 'phr', correspondientes a sintagmas (sintagma nominal, sintagma verbal, etc.).

Como este corpus es ricamente etiquetado se pueden estudiar muchas características, por ejemplo: la categoría morfosintáctica de un sintagma, su función sintáctica o su rol semántico, y combinaciones de ellas.

1.2.1. Evaluación y/o Análisis de Resultados

Para evaluar los resultados obtenidos en esta tesis, se realizaron experimentos con 2 corpus, un corpus en ruso y otro en español, con los cuales se ejecutó el sistema con distintas configuraciones de procesamiento. En el caso del corpus en español se seleccionaron 20 verbos y se compararon las salidas dadas por el sistema con los patrones de subcategorización obtenidos manualmente para ese mismo corpus. En el caso del ruso se analizaron 3 verbos y se compararon los resultados con juicios manuales.

1.3. Descripción del Contenido de la Tesis

En el capítulo 2 se comentarán las diferentes aproximaciones a la adquisición automática de marcos de subcategorización verbales y clases verbales realizadas hasta el momento, dividiéndolas según el tipo de información con que se trabajó. También se explicarán aproximaciones para lenguas distintas del inglés.

Lo que sigue está dividido en dos partes. En la primera parte, en el capítulo 3, se describirán las tareas comunes a todas las ejecuciones del sistema y las particulares que se seleccionarán en el archivo de configuración. También se mostrará la arquitectura del sistema, desde dos puntos de

vista, la arquitectura funcional del sistema y la arquitectura orientada a objetos. Se explicarán los módulos que contienen y se mostrarán los respectivos gráficos. Se mostrará cómo preprocesar un corpus plano para que sea admitido por el sistema, y también se mostrará el diagrama de objetos del sistema, en el cual se podrán ver las relaciones entre los objetos.

En el siguiente capítulo (4) se mencionarán las tecnologías usadas para la flexibilidad del sistema. Se describirá el formato esperado del corpus, que permite distintos formatos definidos por el usuario. También se explicará cómo indicando en el archivo de configuración se podrán solicitar distintas salidas al sistema, dando ejemplos de cómo ejecutar el sistema.

En el capítulo 5 se presentan los detalles de los diferentes módulos del sistema. Se explicará el parser de entrada, cómo se identifican los verbos, patrones y las co-ocurrencias entre ellos. Se describirán los diferentes filtros para establecer o descartar asociaciones entre marcos y verbos: umbral de frecuencias absolutas y relativas y test de hipótesis likelihood ratio. Estos filtros son configurados por el usuario en los parámetros de entrada. También en este capítulo se explicará la estrategia para identificar constituyentes opcionales (también llamados adjuntos). Después de identificar los adjuntos, se podrán colapsar diferentes patrones que comparten los mismos argumentos y sólo difieren en cuanto a los adjuntos.

En la segunda parte presentamos la aplicación del sistema en dos contextos distintos, y la evaluación de los resultados. En el capítulo 6 se comentará las características de un corpus del español etiquetado manualmente, cómo se configuró el sistema, que características lingüísticas fueron estudiadas, y se darán resultados comparando parte de la salida del sistema con la información manual disponible en la base de datos verbal asociada al corpus.

Luego en el capítulo 7 se explica la aplicación del sistema a un corpus plano en ruso, el origen y las cualidades del corpus, prepocesamiento del corpus que incluye información del analizador usado para etiquetar el corpus plano con información morfosintáctica, módulos extra que fueron implementados para construir un XML a partir del corpus, y se muestra el análisis de 3 verbos en ruso que se comparó con juicios humanos.

En 8 se comentará los logros conseguidos y las posibles extensiones del sistema.

Capítulo 2

Trabajo Previo

En este capítulo se comentarán las aproximaciones a la extracción automática de marcos de subcategorización verbal estudiadas, se dividirán los trabajos realizados según los niveles de anotación provista por el corpus, también se explicarán cuáles fueron las aproximaciones realizadas para lenguas distintas del inglés.

2.1. Aproximaciones con poca información

(Brent 1993) trabajó con un corpus en inglés, el Wall Street Journal(WSJ 1994) de 2.6 millones de palabras de texto plano, es decir sin ningún análisis. Para identificar los verbos buscó en el corpus todas las palabras que aparecían con y sin el sufijo 'ing', y luego filtró esa salida con heurísticas de contexto léxico, por ej. no consideró como verbo una palabra si la palabra anterior era un artículo. Identificó los complementos verbales con una gramática de estados finitos, la cual definía por ejemplo patrones lineales como "to-V", el cual se refería a una cláusula infinitiva. Detectó 6 tipos de marcos que contenían objetos directos y cláusulas subcategorizadas e infinitivas, y usó test de hipótesis binomial para determinar la confiabilidad de las asociaciones entre verbos y marcos. Esta aproximación tiene como desventaja que la información adquirida no puede ser extendida para más tipos de marcos, ya que restringe su estudio a las oraciones que pueden ser parseadas con la gramática finita que define. Para evaluar el resultado de su método de adquicisión usó juicios manuales. Su f-score fue de 73.85 %

(Waibel 1993) trabajó con corpus anotados con etiquetas morfosintácticas, de ahí obtuvo información para identificar los verbos, los datos provenían del corpus Wall Street Journal(WSJ 1994), usó 600.000 palabras. Restringió su estudio a 6 tipos de marcos, pero detectó una gran variedad de tipos de complementos y no distinguió entre argumentos y adjuntos. Detectó los tipos de marcos usando una gramática de estados finitos para chunking y expresiones regulares para patrones de chunk lineal. Esta aproximación no se puede extender para otros tipos de marcos.

(Manning 1993) usó 4 millones de palabras del New York Times(Sandhaus), trabajó con corpus analizado sintácticamente, detectó 19 tipos de marcos con información de preposiciones, no distinguió entre argumentos y adjuntos. Usó un parser de estados finitos para parsear las frases que contenían verbos auxiliares (es decir restringe su estudio a frases que tienen auxiliares). Identificó los componentes que seguían al verbo como complementos verbales. Filtró la salida con el test binomial. Para evaluar sus resultados usó la técnica de comparar los tipos de marcos encontrados con tipos de marcos listados en un lexicón existente creado manualmente, el Oxford Dictionary(Hornby and Wehmeier 1985) y reportó un f-score de 58,20 %

(Rooth 1998) usaron 117 millones de palabras del British National Corpus(University 2007). Permitieron todas las combinaciones de verbos y constituyentes adyascentes como tipos de marcos.

Entrenaron una gramática con un algoritmo no supervisado e indujeron información de subcategorización desde el modelo de la gramática entrenada.

También usaron el diccionario Oxford (Hornby and Wehmeier 1985) para la evaluación de sus marcos de subcategorización verbales y reportaron un f-score de 76,95 %

2.2. Aproximaciones con análisis lingüísticos completos automáticos

(John. 1997) usaron 1.2 millones de palabras del corpus Susanne(Sampson 1994), el corpus del inglés hablado SEC(Taylor and Knowles 1988) y el LOB(Lancaster-Oslo/Bergen 1978). Asumieron corpus de datos parcial o totalmente parseados, consideraron 163 tipos de marcos, distinguieron argumentos y adjuntos y dieron una referencia detallada de frases preposicionales, usaron análisis de ranking de salida de un parser probabilístico entrenado con el Penn Treebank(Santorini and Marcinkiewicz 1993) y extrajeron los verbos y los complementos subcategorizados desde el parser, usaron un filtro basado en la semántica de los verbos. Para evaluar sus resultados compararon la salida obtenida con los marcos de subcategorización existentes del Alvey NL tool Dictionary(Boguraev 1987) y el COMLEX sintax dictionary(Grishman/Macleod/Meyers 1994), ellos reportaron un f-score de 46,09 %

2.3. Aproximaciones con análisis lingüísticos completos manuales

(Prolo 2002) hicieron una herramienta para obtener marcos de subcategorización del Penn-Treebank (Santorini and Marcinkiewicz 1993), o para corpus anotados con las convenciones de la anotación del (Santorini and Marcinkiewicz 1993). Ellos examinan cada posible secuencia de etiquetas, ambas funcionales y categóricas y determinan cuando tal secuencia indica un argumento obligatorio, un argumento opcional o un modificador. Proveen reglas de fina granularidad para analizar el corpus.

(O'Donovan Ruth/Michael Burke/Aoife Cahill 2005) realizaron adquisición de marcos de subcategorización sintácticos basados en una gramática derivaron sus verbos e información de marcos de los que salieron mal en anotación automática del (Santorini and Marcinkiewicz 1993). Adquirieron información de marcos con y sin información de preposiciones, distinguieron marcos activos y pasivos. Ellos no predefinieron el conjunto de marcos. Evaluaron sus resultados comparándolos con información del (Grishman/Macleod/Meyers 1994) y obtuvieron un f-score de 26,3 % con frases preposicionales y 64,3 % sin frases preposicionales.

2.4. Aproximaciones para lenguas distintas del inglés

(Eckle-Kohler 1999) adquirieron información de marcos de subcategorización verbal sintácticos para 6305 verbos en alemán a partir de un corpus etiquetado morfosintácticamente, usaron heurísticas lingüísticas definidas por consultas a expresiones regulares sobre el uso de 244 tipos de marcos incluyendo frases preposicionales.

También para el alemán, (Wauschkuhn 1999) trabajó con corpus anotado con etiquetas morfosintácticas, extrajo un máximo de 2000 oraciones por cada verbo, con un total de 1044 verbos y construyó una gramática libre de contexto para análisis parcial. Con el análisis sintáctico consiguió patrones de valencias, luego los agrupó con el fin de extraer las combinaciones más frecuentes de patrones y verbos, encontró 42 tipos de marcos. Para evaluar sus resultados eligió juramentos

manuales de 7 verbos, reportando un f-score de 61,86 %.

(Schulte im Walde 2000) trabajó con 18.7 millones de palabras de corpus de diarios alemanes, consiguió marcos de subcategorización para más de 14000 verbos para 38 tipos de marcos puramente sintánticos y un refinamiento de 178 tipos de marcos incluyendo distinciones de frases preposicionales. Desarrolló una gramática libre de contexto conteniendo reglas de la gramática (reglas que describen el uso del idioma, tienen información morfológica, sintáctica y otras) y usó un ambiente de entrenamiento no supervisado de encabezados lexicalizados probabilísticos de las gramáticas libres de contexto HL-PCFG para entrenar la gramática del corpus. Usó el diccionario Duden das Stilwörterbuch(AG 2001) para evaluar sus resultados, obtuvo un f-score de 57,24 % con frases preposicionales y 62,30 % sin frases preposicionales.

Para el portugués, (de Lima 2002), y para el griego, (Georgala 2003), usaron un ambiente no supervizado de HL-PCFG para entrenar la gramática del corpus y de ahí obtener las combinaciones de verbos y marcos para los verbos del corpus.

Para el checo, (Zeman 2000) usaron las definiciones de dependencia sintáctica del Prague Dependency Treebank(Sgall 2001) para inducir marcos de subcategorización verbal. Ellos usaron un algoritmo de aprendizaje para distinguir mejor entre argumentos y adjuntos. Obtuvieron 88 % de precisión en distinguir argumentos de adjuntos sobre textos parseados no vistos, es decir con textos no usados como material de entrenamiento. Para filtrar sus salidas usaron test de hipótesis Likelihood Ratio, T-score y binomial, y obtuvieron una precisión del 82 %, 82 % y 88 % respectivamente, y recall de 77 %, 77 % y 74 % respectivamente.

Para holandés (Heid 2003) desarrollaron un chunker recursivo para textos no restringidos para extraer subcategorización verbal. Primero identificaban pequeños chunks y por aplicación iterativa de reglas simples analizaban los casos donde tales pequeños chunks eran partes de otros más largos. Hacen énfasis en frases sustantivas.

Para el francés, (Salmon-Alt 2006) crearon un corpus con ocurrencias aleatorias de 104 verbos frecuentes del Frantext online literaly database (de la Langue Française), obtuvieron 27 marcos de subcategorización como cualquier combinación de un conjunto restringido de componentes (objetos directos, frases preposicionales pre-especificadas, claúsulas, frases adjetivas etc.)

Para el español (Chrupala 2003) adquirió marcos de subcategorización para los verbos del SenSem (Fernández 2004) , estudio alternación diátesis, y clases verbales, filtró sus resultados con test de hipótesis Binomial, frecuencia relativa, y obtuvo un f-score de 69 % y 74 % respectivamente.

Parte I

Sistema

Capítulo 3

Arquitectura del Sistema

En este capítulo presentamos la arquitectura del sistema. Como hemos dicho, uno de los objetivos del sistema es una alta flexibilidad, para poder tratar textos en diferentes lenguas, con diferentes tipos de análisis lingüístico y con diferentes objetivos. El sistema se diseñó con este objetivo como prioridad.

Primero vamos a detallar las tareas centrales del sistema, que se ejecutan independientemente de los parámetros de configuración, y las tareas opcionales especificadas en el archivo de configuración.

Luego se muestra la arquitectura desde dos puntos de vista, la arquitectura funcional y la arquitectura orientada a objetos, se explicará la funcionalidad de cada uno de los módulos que conforman dichas arquitecturas y se mostrarán gráficos para ambas.

También se mostrará el diagrama de objetos y se explicará como los objetos se interrelacionan.

En el siguiente capítulo se mostrará cual es la entrada del sistema, la salida del sistema, cómo ejecutar el sistema, y cómo configurarlo para las distintas opciones de procesamiento.

También se explicarán las opciones por defecto que provee el sistema.

Luego en el capítulo 5 se presentan los detalles de los diferentes módulos del sistema.

3.1. Tareas comunes a todas las aproximaciones

Las siguientes tareas son las que el sistema siempre ejecutará sin importar por ejemplo si el corpus es plano, o ricamente etiquetado.

- Parsear parámetros provistos por el usuario.
- Comprobar correctitud y compatibilidad de parámetros o dar el correspondiente mensaje de error en caso de no ser la cantidad o tipos de datos esperados por el sistema.
- Seleccionar verbos de trabajo, es decir, ver si el usuario eligió trabajar sólo con una lista de verbos o con todos los verbos que se encuentren en el corpus.
- Obtener patrones de las oraciones del corpus de entrada, para cada una de las características y/o combinaciones de características incluyendo la forma, según lo seleccionado por el usuario en el archivo de configuración. Para ello son necesarios los siguientes pasos:
  1. Identificar los verbos.
  2. Identificar el contexto de cada verbo.

3. Para cada característica, o combinación de características, obtener patrón en el alcance elegido por el usuario, o por defecto en una longitud de 2 palabras a cada lado del verbo.

- Asociar verbos y patrones, es decir, los patrones asociados con éxito pasan a tener el estatus de marco de subcategorización para un determinado verbo. De esta manera se crea el diccionario. Para crear o agregar una entrada en el diccionario de salida, la asociación entre el verbo y el patrón deberá pasar los siguientes tests, según fueron seleccionados en el archivo de configuración:

  1. Umbral de frecuencia absoluta, mínima cantidad de ocurrencias del verbo para poder considerarlo.
  2. Umbral de frecuencia relativa, mínima cantidad de co-ocurrencia de un patrón con un verbo para considerarlo.
  3. Test de hipótesis Likelihood Ratio.

3.2. Tareas opcionales seleccionadas en la configuración

Las siguientes son las tareas opcionales que el usuario podrá requerir al sistema, por medio del archivo de configuración

- Se inicializará el sistema con un diccionario de subcategorización preexistente, en este caso se actualizarán las entradas de los verbos en dicho diccionario y se agregarán las entradas que no estén en el diccionario dado y que provengan del procesamiento del corpus provisto por el usuario. En caso de tener lista de verbos a considerar no se eliminarán las entradas del diccionario dado por más que haya verbos que no estan en la nueva lista de verbos. Notar que restaurando un diccionario, podría ejecutar el sistema con un corpus vacío para filtrar el diccionario existente, o colapsar patrones.
- Se podrá usar la forma de las palabras como constituyentes de los patrones si esta opción esta seleccionada en el archivo de configuración.
- Tener en cuenta o ignorar el orden de las etiquetas para formar el patrón, según lo indicado en el archivo de configuración.
  En caso de ignorar el orden, se tomará el orden alfabético de los elementos del patrón.
- Rellenar los patrones con una palabra en particular o no rellenar.
- Agregar una marca verbal en la posición del verbo, o no agregarla si no fue seleccionado en el archivo de configuración.
- Considerar solo los verbos provenientes de una lista de verbos. Es decir se tendrán en cuenta sólo estos verbos para la búsqueda de marcos de subcategorización y se ignorarán los demás, o considerar todos los verbos encontrados en el corpus.
- Colapsar patrones, a fin de identificar constituyentes opcionales y con esto se agrupen patrones que quizás otros test del sistema filtrarían. Al colapsar patrones perderemos la marca verbal, los símbolos para de relleno de patrones, y el orden.

3.3. Arquitectura Funcional del Sistema

En este capítulo se mostrará la arquitectura funcional del sistema, se explicará los pasos previos que deberán realizarse a los textos con diferentes tipos de anotación desde nula hasta anotaciones muy ricas (corpus etiquetados), para luego ejecutar el sistema, y con ello adquirir los marcos de subcategorización verbal para todos los verbos del corpus o los seleccionados en una lista de verbos.

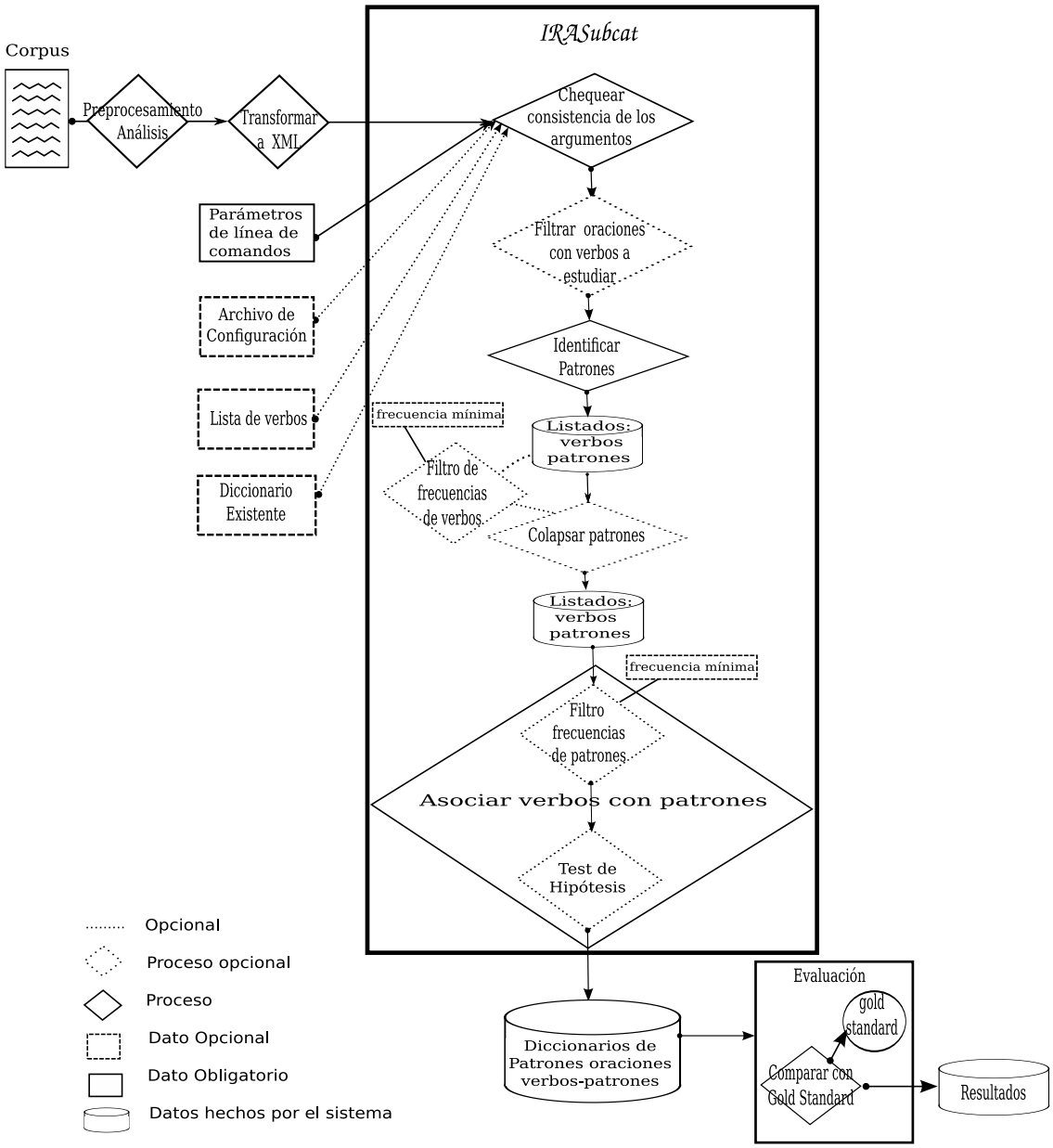

Figura 3.1: Arquitectura Funcional del Sistema

3.3.1. Preprocesamiento de corpus

Este paso no se incluye en el sistema, pero es habitual realizarlo en estas tareas. El data sparseness es el esparcimiento se los datos, el análisis reduce el data sparseness porque da abstracción, sobre todo con sintagmas, luego veremos este problema cuando analicemos la ejecución del corpus ruso que no posee información de sintagmas.

Para los corpus planos, es decir, para los textos que no posean ningún tipo de información asociada a los items léxicos, habrá que preprocesarlos a fin de detectar y agregar información morfosintáctica a los verbos antes de ejecutar el sistema. Para realizar esto hay varias opciones, una opción es preprocesar el corpus con heurísticas para encontrar verbos en la lengua particular, como realizó (Brent 1993) para el inglés.

Otra opción es buscar si hay disponible algún etiquetador morfosintáctico, con el se pueda adquirir información morfosintáctica para las palabras del idioma estudiado.

También se necesitará un módulo que tome el corpus de salida del paso anterior y con éste construya un archivo XML bien formado codificado en UTF-8. El módulo que construye el XML deberá detectar las oraciones en el corpus y colocar las respectivas etiquetas de inicio y fin de oración. También deberá poner el inicio y fin del corpus.

Por ejemplo, si la salida del etiquetador es la palabra, la etiqueta morfosintáctica de la palabra y el lema de la palabra

```
'Liliana' NP <unknown>
'compra' VLfin comprar
'frutas' NC fruta
```

entonces el archivo XML de la oración mencionada que recibe el sistema es el siguiente:

```
<corpus>
   <oracion>
      <palabra lema='unknown' morfosint='NP'>Liliana</palabra>
      <palabra lema='comprar' morfosint='VLfin'>compra</palabra>
      <palabra lema='fruta' morfosint='NC'>frutas</palabra>
   </oracion>
</corpus>
```

El sistema requiere que el usuario provea un corpus con buena formación de XML. Para ser bien formado el XML deberá cumplir las siguientes condiciones:

- Los documentos han de seguir una estructura estrictamente jerárquica en lo que respecta a las etiquetas que delimitan sus elementos. Una etiqueta debe estar correctamente incluida en otra, es decir, las etiquetas deben estar correctamente anidadas. Los elementos con contenido deben estar correctamente cerrados.
- Los documentos XML sólo permiten un elemento raíz del que todos los demás sean parte, es decir, solo pueden tener un elemento inicial.
- Los valores de los atributos en XML siempre deben estar encerrados entre comillas simples o dobles.
- El XML es sensible a mayúsculas y minúsculas. Existe un conjunto de caracteres llamados espacios en blanco (espacios, tabuladores, retornos de carro, saltos de línea) que los procesadores XML tratan de forma diferente en el marcado XML.
- Es necesario asignar nombres a las estructuras, tipos de elementos, entidades, elementos particulares, etc. En XML los nombres tienen alguna característica en común.

3.3.2. Flujo de procesamiento

1. **Chequear consistencia de los argumentos**: este proceso toma como entrada los parámetros provistos en línea de comandos.
   Se encargará de chequear la consistencia, la cantidad de parámetros, la forma de los parámetros y en el caso de los parámetros que refieren a archivos, la existencia de los mismos, para garantizar que el sistema podrá ejecutar.

2. **Identificar oraciones en el corpus**: este proceso toma como entrada el corpus y da oraciones del mismo.

3. **Filtrar oraciones que no tienen verbos que se consideran**: este proceso se encarga de pasar al proceso identificador de patrones (el siguiente), las oraciones que posean verbos que se tienen en cuenta. La estructura de datos que devuelve este proceso, se usa en el siguiente proceso.

4. **Identificar patrones**: este proceso toma una oración, un verbo a considerar, una lista de características a considerar y un alcance a tener en cuenta y busca los patrones de esas características alrededor del verbo considerado, en el alcance seleccionado por el usuario o en el alcance proporcionado por defecto por el sistema. Esto se explicará con más detalle en el capítulo 5

5. **Colapsar patrones**: este proceso identificará constituyentes opcionales por verbo y colapsará los patrones que difieran sólo en esos constituyentes identificados como opcionales.

6. **Asociar verbos con patrones**: este proceso toma un verbo, con su frecuencia de ocurrencia en el corpus, un patrón y su frecuencia de co-ocurrencia con ese verbo y la frecuencia absoluta de ocurrencia del patrón en el corpus. Luego ejecuta test de frecuencia absoluta para la cantidad de ocurrencia del verbo, test de frecuencia relativa para la cantidad de co-ocurrencias del patrón con ese verbo y test de hipótesis likelihood ratio para ver la confiabilidad de la relación entre el verbo y el patrón. Esto último se verá reflejado en el diccionario de salida como se explicará en el capítulo 4.

3.4. Arquitectura Orientada a Objetos del Sistema

La arquitectura orientada a objetos del sistema está compuesta por los siguientes módulos que son clases en la implementación:

Clase interfaz con el usuario: brinda funcionalidad para ejecutar el sistema y mostrar los mensajes de error.

Clase chequear Precondición: esta clase provee funcionalidad para chequear cantidad, tipos y existencia de los parámetros ingresados por el usuario, así como también la compatibilidad entre los parámetros.

Clase Parámetros de configuración: esta clase provee funciones para parsear el archivo de configuración y extraer los parámetros de configuración.

Clase Corpus: Esta clase incluye las funciones para crear el corpus a partir de un archivo XML válido, construir oraciones a partir de lo que va extrayendo del corpus, dar oraciones, y ver

si hay más oraciones.

Clase Diccionario: Aquí están las funciones para crear un diccionario desde un archivo XML, y crear un diccionario vacío, agregar una entrada al diccionario, actualizar una entrada existente, ver si un verbo está entre las entradas del diccionario.

Clase Oración: En esta clase están las funciones para extraer partes de la oración, es decir elementos XML que representan palabras y características lingüísticas mediante los atributos.

Clase Palabra: En esta clase están las funciones para crear una palabra, que puede contener la forma de la palabra y las características de la misma y extraer forma o características de la palabra.

Clase Verbo: Esta es una subclase de la clase anterior, son palabras especiales que tienen la característica de ser verbos.

Clase Patrón: sirve para crear patrones desde una lista de constituyentes, modificar atributo de constituyentes opcionales, extraer lista de constituyentes, y extraer constituyentes opcionales.

Clase Trie: Esta clase provee funcionalidad para crear y agregar elementos a una estructura de datos de árbol trie. También provee métodos para recuperar el número de ocurrencias de un patrón, preguntar si existe en patrón en la Trie y si es una hoja de la Trie.

Clase Extractor de patrones: Esta clase toma un diccionario y una oración busca el patrón, para cada característica o combinaciones de ellas, y/o items léxicos, de la oración y actualiza un diccionario intermedio (ya que no es el diccionario final) con ese patrón.

Clase Diccionario de patrones y contadores: Esta clase provee funciones para crear un patrón con un contador, actualizar el contador de un patrón y extraer el número de veces que ocurrió el patrón.

Clase Diccionario de etiquetas y objetos de la clase anterior: Esta clase provee funciones para crear un diccionario de etiquetas con objetos que pertenecen a la clase anterior, es decir son diccionarios de patrones con contadores, también provee funcionalidad para actualizar una entrada del diccionario y para extraer una entrada del diccionario.

Clase Colapsador de patrones: Aquí se identifican constituyentes opcionales para cada verbo y se colapsan los patrones según condiciones que se explicarán en el capítulo 5

Clase Test de hipótesis: provee funciones para que dado un verbo, un patrón que ocurre con ese verbo, y la frecuencia de ocurrencia del patrón con el verbo, calcular si la asociación entre el verbo y el patrón es confiable tomando como umbrales del test a 0.90, 0.95 0.99 y 0.995.

En el diagrama 3.2 se muestra la forma en que se interrelacionan los objetos en el sistema, y se pueden ver que:

- Los parámetros del sistema son tanto los parámetros de línea de comando, como el archivo de configuración, el diccionario existente y la lista verbal si los hubiera.
- Un objeto corpus está compuesto por objetos oraciones, y éstas a su vez están compuestas por uno o más objetos palabra y posiblemente por uno o más objetos de la subclase de palabra "verbo".
- El proceso control toma los parámetros de línea de comandos, del objeto corpus y la configuración, envía las oraciones al Extractor, el cual retorna la información del patrón para

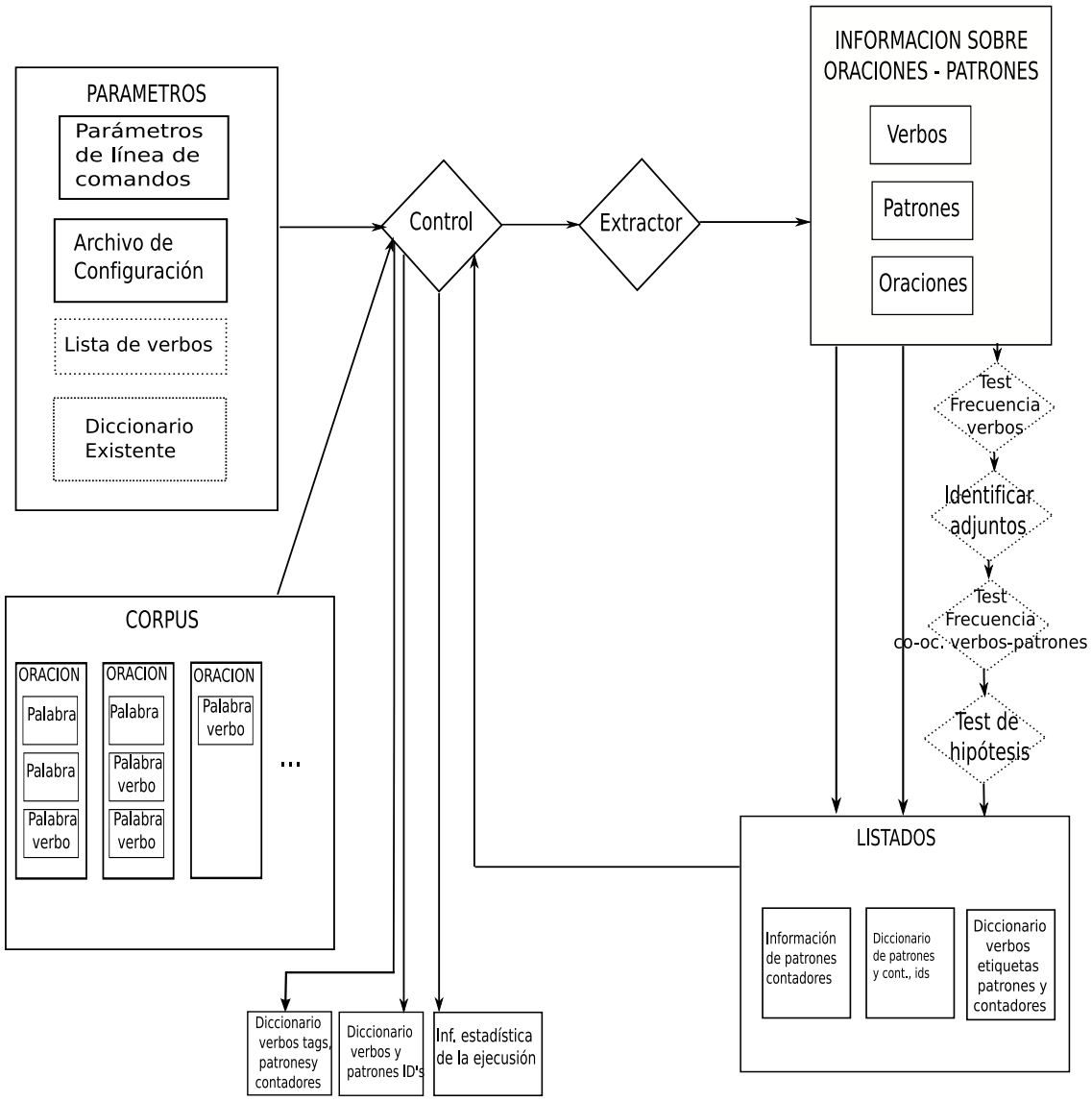

Figura 3.2: Diagrama de Objetos del Sistema.

la oración dada, esta información actualiza un listado, por verbo y etiqueta, que luego será enviado al colapsador, si así lo indica el archivo de configuración, y luego a los tests del sistema, y por último se creará el diccionario de subcategorización de salida. También se creará un diccionario con ID's de oraciones que fueron ejemplos de cada patrón, si el corpus proveía esta información como atributos del ámbito padre del cual se buscaron los constituyentes de los patrones. Y por último un archivo 'info_file' con información estadística del procesamiento realizado.

Capítulo 4

Entrada y Salida del Sistema

En este capítulo se explicarán las medidas tomadas para que el sistema se pueda ejecutar con diferentes niveles de etiquetado para el corpus de entrada, con diferentes opciones de procesamiento, como configurar el sistema para obtener diferentes salidas o actualizar un diccionario existente con la ejecución del sistema.

El sistema toma como entrada una serie de parámetros de línea de comando y opcionalmente un archivo de configuración, una lista de verbos a considerar y un diccionario existente. Todos los archivos nombrados deben estar codificados en UTF-8.
Para la flexibilidad del sistema se usaron las siguientes tecnologías:

El corpus de entrada al sistema deberá estar en formato XML. Se eligió XML para el formato del corpus de entrada del sistema por ser XML un lenguaje extensibles de marcas, que proporciona un modo consistente y preciso de aplicar etiquetas para describir las partes que componen un texto, permitiendo además el intercambio de documentos entre diferentes plataformas.

Notar que un corpus que no contenga atributos para ninguna palabra será un corpus plano. El sistema realizado en esta tesis, necesita como mínimo que estén etiquetados las funciones verbales, de las cuales se estudiarían los marcos de subcategorización léxicos.

Como el sistema realizado en esta tesis es independiente de lengua, se necesitó una forma de dar al sistema información en distintos idiomas, con una sola codificación. Por este motivo, el sistema requiere que el corpus de entrada, y si hubiera archivo de configuración corpus existente y lista verbal, que todos estén codificados en UTF-8. La sigla UTF-8 la cual significa formato de transformación de unicode a 8 bits (8-bit Unicode Transformation Format) es una norma de transmisión de longitud variable para caracteres codificados utilizando Unicode, creada por Rob Pike y Ken Thompson. UTF-8 usa grupos de bytes para representar el estándar de Unicode para los alfabetos de muchos de los lenguajes del mundo.

De esta manera, el sistema acepta corpus en ruso, en español, en inglés, etc, si esos corpus están codificados en UTF-8.

El sistema tomará un corpus haciendo abstracción de los nombres de las etiquetas y los nombres de los atributos que usemos (por ejemplo en vez de 'corpus' poner 'information' y en vez de 'palabra' poner 'word') y para el caso de los atributos podríamos tener un corpus en el cual el atributo que contiene la información sobre categoría morfosintáctica de las palabras se llame 'sint' y otro corpus en el que sea 'sintactic'. El sistema acepta que el corpus de entrada tenga un anidamiento más profundo, siempre y cuando la etiqueta en la cual se buscan los patrones sea la etiqueta padre de las etiquetas que contienen la información de las palabras. El sistema necesita que se especifiquen completamente los finales de las etiquetas, es decir si una etiqueta es '<frase>', el final de la

etiqueta deberá ser '</frase>', y no está permitido que el final sea '/>' porque para implementar el sistema se usó una librería para procesamiento de XML que no lo permite.

4.1. Archivo de Configuración y opciones por defecto

Se podrán comentar líneas del archivo de configuración anteponiendo el caracter '#'. En caso de estar comentada una línea, el sistema tomará la opción por defecto.

Como se mencionó anteriormente lo que puede faltar es el archivo de configuración. En cuyo caso se ejecutará el sistema con las opciones por defecto, que mostraremos en las sucesivas líneas.

TO CONSIDER VERB LIST = 100_verbos_Mas_Frec.txt
Las opciones son 'NO' y 'nombre_de_archivo'. Esta opción permite que el sistema considere sólo los verbos que figuran en una lista verbal, en este caso el nombre del archivo en el cual están los verbos a ser estudiados se llama "100_verbos_Mas_Frec.txt", el sistema requiere que este archivo esté codificado en UTF-8, este archivo debe estar situado en la misma carpeta que IRASubcat.
La forma del archivo será la siguiente:
verbo1
verbo2
verbo3
...
Es decir, los verbos están separados entre sí por la techa "Enter".
El suario también podría querer considerar todos los verbos del corpus, en cuyo caso basta poner esta opción en "NO", y el sistema se ejecutará teniendo en cuenta todos los verbos que encuentre según el patrón de búsqueda que indica que una palabra es un verbo.

Por defecto:

- No se considerará lista de verbos, se considerarán todos los verbos que se encuentren en el corpus.

DICTIONARY EXIST = NO
Si el usuario lo que desea es actualizar un diccionario existente, o filtrar las entradas, o colapsar las entradas, debería poner aquí el nombre del diccionario, por ejemplo si el diccionario a actualizar se llama "Diccionario_Sp.xml", ese sería lo que debería poner en esta opción. Cabe recordar que el diccionario existente debe ser un archivo codificado en UTF-8 con extensión XML, que el XML sea válido y que respete la DTD del diccionario de salida del sistema propuesto en esta tesis. Dicho archivo deberá estar situado en la misma carpeta que IRASubcat

Por defecto:

- No se considerará ningún diccionario existente.

LONG OF SIDE THE VERB FOR PATTERN = ALL
Esta opción permite al usuario del sistema, tener en cuenta distintos ámbitos en los cuales adquirir los patrones de las oraciones. Aquí el usuario podrá poner un número por ejemplo 4, el cual significará que se buscan 4 etiquetas a cada lado del verbo. Si el usuario pone en esta opción ALL significará que las etiquetas que se buscan son todas las que están en el ámbito de la oración.

Por defecto:

- Se considerará el ámbito a cada lado del verbo como 2.

COMPLETE WITH WORD = NO
Esta opción lo que permite es que el sistema complete los patrones encontrados con una palabra particular, por ejemplo si en esta opción lo que ponemos es NONE, y si en la opción anterior

teníamos el número 3, el sistema completará ambos lados del patrón encontrado, hasta llegar a la longitud de 6, los cuales están distribuidos 3 a la derecha del verbo y 3 a la izquierda, en el caso que en la oración no haya 3 etiquetas a cada lado del verbo.

Por defecto:

- No se completará el patrón encontrado con ninguna palabra.

ORDER OFF TAGS = YES
Las opciones aquí son "YES" y "NO", las cuales significan que se tomará en cuenta o no el orden de las etiquetas encontradas en las oraciones. Cabe destacar que si no tenemos en cuenta el orden, tendremos menos patrones, esta es una forma de agrupar datos, y así reducimos el data-sparseness ya que se agrupan los patrones que tienen los mismos componentes pero en distintas ubicaciones, para realizar esto el sistema ordena los componentes alfabéticamente. Para seleccionar este parámetro adecuadamente se debería tener en cuenta la estructura del idioma bajo estudio.

Por defecto:

- No se considerará el orden de las palabras.

TARGET TAGS = sint, lema
En este parámetro de configuración ponemos separado por "," todas las etiquetas que queremos estudiar, esta opción puede ser "NO", por ejemplo si sólo estudiamos los items léxicos.

Por defecto:

- Se considerará 'sint'.

USE LEXICAL ITEMS = NO
Este parámetro admite las opciones "YES" y "NO", si se estudian los items léxicos o si no respectivamente. Notar que esta opción en "YES", nos da información muy detallada, de las palabras que van alrededor de los verbos, pero el diccionario de salida será de mucho mayor tamaño que si hubieramos considerado el lema o la categoría morfosintáctica por ejemplo, ya que los items léxicos tienen más variabilidad.

Por defecto:

- No se considerarán las formas canónicas de las palabras.

INTRODUCE VERBAL MARK = NO
Esta opción también admite "YES" y "NO", y en los casos "YES" el sistema pondrá una marca en la posición que encontró el verbo, la marca será el símbolo '|'.

Por defecto:

- No se introducirá marca verbal.

COLAPSE PATTERNS = NO
El sistema puede colapsar patrones, para ello identifica constituyentes opcionales, si esta opción está en 'YES'.

Por defecto:

- No se colapsarán patrones.

MAX ITERATION FOR FIND COLAPSE PATTERNS = FALSE
Es la cantidad de iteraciones que realizará el sistema tratando de colapsar patrones. Esta opción admite un número o FALSE, en el caso de ser un número hará tantas iteraciones como indique

el número, o menos en el caso de no ver progreso, y en el caso de ser FALSE, hará todas las iteraciones que pueda, hasta no ver progreso. (llamamos progreso a que en una iteración identifique al menos 1 constituyente opcional)

Por defecto:

- Al no colapsarse los patrones por defecto, este item no se tiene en cuenta.

MINIMAL ABSOLUTE VERBAL FREQUENCY = 0
Esta opción corresponde al nivel del test de frecuencia absoluta del verbo, es decir, en este caso al tener el número '0', la opción esta desactivada, pero si en esta opción ponemos 10, el diccionario de salida del sistema contendrá solo los verbos que ocurran como mínimo 10 veces, y además hayan pasado los otros test.

Por defecto:

- La mínima cantidad será 0, es decir no se descartarán verbos por ocurrir pocas veces.

MINIMAL RELATIVE PATTERN FREQUENCY = 0
Esta opción es como la anterior pero considera la co-ocurrencia del patrón con el verbo, y si este número fuera 10, el patrón deberá ocurrir como mínimo 10 veces con el verbo, y pasar los demás test, para aparecer el diccionario de salida del sistema.

Por defecto:

- La mínima cantidad será 0, es decir no se descartarán co-ocurrencias de verbos con patrones por ocurrir pocas veces.

USE LIKELIHOOD RATIO TEST = YES
Si usará test Likelihood Ratio para filtrar co-ocurrencias.

Por defecto:

- Se usará test likelihood ratio para filtrar las co-ocurrencias. Se dará como resultado la certeza de que el patrón pertenezca al marco de subcategorización del verbo, y cuyas opciones son: 90 %, 95 %, 99 %, 99.5 %, "no_paso" en caso no haber pasado ningún umbral y 'NO DECIDE' en caso de no cumplir las condiciones para ejecutar el test.

4.2. Especificación de la Salida del sistema

El sistema dará como salida un diccionario de subcategorización verbal, un diccionario con información de los identificadores de las oraciones (si hubiera ID's en las oraciones del corpus a procesar), y un archivo de información el cual contendrá datos estadísticos e información de tiempo empleado, respecto al procesamiento solicitado.

El diccionario de subcategorización verbal será un archivo XML, codificado en UTF-8 con la siguiente información:

- En la raíz del diccionario de salida estarán los atributos de configuración mediante los cuales se obtuvo dicho diccionario.
- El diccionario contendrá por cada entrada un atributo obligatorio (el verbo), un contador de ocurrencias del verbo en el corpus y una lista de patrones con sus respectivas cantidades con el verbo y cantidad de ocurrencias en todo el corpus. Cabe destacar que las entradas que figuren en el diccionario serán todas las entradas resultantes de la ejecución del sistema con los parámetros provistos o los parámetros por defecto del sistema, los verbos que no superen

la mínima cantidad prevista, no estarán en el diccionario de salida, al igual que los patrones, en cambio se darán todas las salidas del test de hipótesis likelihood ratio, si se usó este test. Todas las salidas del diccionario estarán ordenadas de mayor a menor según la cantidad de ocurrencia de los verbos y de los patrones.

Ejemplo:

Supongamos que en el corpus hay oraciones con el verbo 'amar', y que aparecen el 80 % de las veces con el patrón [Sust,|,Prep,Sust] y el 20 % de las veces con el patrón [Sust,|,Verbo], ejemplos de nombrados patrones serían: para el primer patrón la oración "Andres ama a María" y para el segundo patrón "Andres ama cantar", entonces la salida del sistema tendrá la siguiente forma:

```
<dictionary
   execute="IRASubcat.py nombre_corpus.xml cat=verbo s lema_verbo configSp.txt
   verb_list="False" dict_exist="False" scope="ALL" comp_w_word="False"
   order="True" tags="['sint']" lex_items="False" verbal_mark="True"
   verb_min_abs_freq="0" pattern_min_rel_freq="0" use_likelihood="False">
   <entry verb='amar'>
      <tag name='sint'>
         <pattern id='[Sust,|,Prep,Sust]' count_w_verb='8' total_count='12'>
         </pattern>
         <pattern id='[Sust,|,Verbo]' count_w_verb='2' total_count='5'>
         </pattern>
      <tag>
      ...
   </entry>
   ...
</dictionary>
```

Los atributos que aparecen en la etiqueta dictionary, corresponden a la forma de ejecución y a los parámetros con los cuales se ejecutó el sistema. Ahora mostraremos el formato general de la DTD que sigue el dicccionario de salida y a continuación se explicará cada una de las líneas que la componen.

```
<!ELEMENT dictionary (entry)*>
<!ELEMENT entry (tag)+>
<!ELEMENT tag (pattern)+>
<!ELEMENT pattern (#PCDATA)>
<!ATTLIST dictionary execute CDATA #REQUIRED>
<!ATTLIST dictionary verb_list CDATA #REQUIRED>
<!ATTLIST dictionary dict_exist CDATA #REQUIRED>
<!ATTLIST dictionary scope CDATA #REQUIRED>
<!ATTLIST dictionary comp_w_word CDATA #REQUIRED>
<!ATTLIST dictionary order CDATA #REQUIRED>
<!ATTLIST dictionary tags CDATA #REQUIRED>
<!ATTLIST dictionary lex_items #REQUIRED>
<!ATTLIST dictionary verbal_mark CDATA #REQUIRED>
<!ATTLIST dictionary verb_min_abs_freq CDATA #REQUIRED>
<!ATTLIST dictionary pattern_min_rel_freq CDATA #REQUIRED>
<!ATTLIST dictionary use_likelihood #REQUIRED>
<!ATTLIST entry verb CDATA #REQUIRED>
<!ATTLIST tag name CDATA #REQUIRED>
<!ATTLIST pattern count_w_verb CDATA #REQUIRED>
<!ATTLIST pattern total_count CDATA #REQUIRED>
```

La primera línea especifica que la etiqueta raíz del archivo XML de salida será "dictionary", y que estará compuesta por 0 o más etiquetas "entry", es decir estará compuesto por 0 o más entradas. En la segunda línea se establece que cada entrada tendrá al menos una etiqueta (tag), en la tercera cada etiqueta al menos un patrón, la cuarta línea quiere decir que el patrón es un dato. Desde la quinta hasta la decimo sexta se especifican los atributos del diccionario, los cuales son básicamente las opciones de ejecución del sistema, para obtener dicho diccionario. Desde la decimo séptima en adelante se especifica la lista de atributos para la entrada, el tag y para el patrón. Para la entrada un atributo obligatorio es el verbo, para el tag, el nombre (name), en el ejemplo la etiqueta 'sint' que especifica la etiqueta morfosintáctica, y para el patrón el atributo es la cantidad de co-ocurrencias con el verbo.

En caso de actualizar un diccionario existente, el diccionario existente tendrá que ser validado con la DTD nombrada anteriormente.

El diccionario de ID's será un archivo codificado en UTF-8 con el siguiente formato:

```
<!ELEMENT dictionary (entry)*>
<!ELEMENT entry (tag)+>
<!ELEMENT tag (pattern)+>
<!ELEMENT pattern (#PCDATA)>
<!ELEMENT s_list (#PCDATA)>
<!ATTLIST entry verb CDATA #REQUIRED>
<!ATTLIST tag name CDATA #REQUIRED>
<!ATTLIST pattern id CDATA #IMPLIED>
```

El 'id' del patrón estará dado de la siguiente manera

```
[NC]{A:3 ART:2}
```

Siendo los constituyentes que figuran dentro de los [ ] los argumentos del verbo, y los que figuran entre {}, los constituyentes que encontró el sistema como adjuntos. Notar que si no se colapsan patrones, no aparecerán constituyentes entre llaves.

y el archivo de estadísticas contendrá la siguiente información:

- Tiempo empleado en la ejecución del sistema.
- Cantidad total de verbos considerados.
- Cantidad total de patrones encontrados por etiqueta.
- Cantidad de verbos rechazados por el test de frecuencias.
- Cantidad de patrones rechazados por el test de frecuencias.
- Cantidad de patrones rechazados por el test Likelihood Ratio.
- Ranking de constituyentes por verbo y por etiqueta estudiada. (quizas esto sea útil para encontrar adjuntos)

4.3. Ejemplo de Ejecución

El sistema se ejecuta por línea de comandos.

Es necesario tener python instalado, la librería xml.parsers.expat, que es la que se usó para parsear el XML del corpus de entrada.

Es necesario tener el corpus que se desea procesar en el mismo directorio que **IRASubcat.py**, así como también el diccionario existente, el archivo de configuración y la lista verbal si hubiera.

Supongamos que el nombre del corpus a procesar es "corpus.xml", y que el ámbito en el que el sistema va a buscar las palabras en el corpus es "oracion", es decir hay etiquetas oracion en el corpus cuyos hijos directos son "palabra", y que palabra contiene los atributos o las palabras canónicas que serán componentes de los patrones de subcategorización que encontrará el sistema.

Tenemos que identificar en nuestro corpus cual es la etiqueta y el valor que indican que una palabra es un verbo, ya que es requisito del sistema proveer la etiqueta y el valor verbal. Supongamos que la etiqueta es "morfosint" y el valor es "V".

Será necesario indicarle al sistema si las entradas del diccionario de salida serán las palabras, es decir la forma de la palabra, en cuyo caso pondremos en ese parámetro "lexical" (lexical es una palabra reservada del sistema), o de lo contrario una etiqueta cuyos valores serán usados como entradas en el diccionario de subcategorización de salida. Como ejemplo y sugerencia de esta última etiqueta podríamos usar la etiqueta "lema" o la que represente el lema de la palabra, si el corpus contara con esa información.

Entonces una de las posibilidades es ejecutar el sistema sin archivo de configuración con lo cual el sistema se ejecutaría teniendo en cuenta las opciones por defecto provistas por el sistema, para ejecutar el sistema de este modo escribiremos en línea de comandos:

python IRASubcat.py corpus.xml morfosint=V oracion lexical
en el caso de usar la forma de la palabra como entradas del diccionario de salida
o
python IRASubcat.py corpus.xml morfosint=V oracion lema
en caso de usar el lema de las palabras como entradas en el diccionario de salida

Otra de las posibilidades de ejecutar el sistema es con un archivo de configuración, supongamos que el archivo de configuración tenga como nombre config.cfg, en este caso se ejecutará el sistema de la siguiente manera:

python IRASubcat.py corpus.xml morfosint=V oracion lexical config.cfg
o
python IRASubcat.py corpus.xml morfosint=V oracion lema config.cfg

Ahora introducimos un ejemplo completo para ilustrar las distintas opciones:
Supongamos que contamos con un corpus cuya única oración es "Caminar es bueno para la salud", en este caso es un corpus plano, en español, al cual procesaremos con un etiquetador morfosintáctico el TreeTagger (Schmid 1994), obteniendo lo siguiente:

```
Caminar VLinf caminar
es      VLfin ser
bueno   ADJ   bueno
para    PREP  para
la      ART   el
salud   NC    salud
```

Según el conjunto de etiquetas (Tagset) del TreeTagger (Schmid 1994), en la primera línea la etiqueta que le corresponde a "Caminar" es "VLinf", la cual significa verbo clítico infinitivo, y "caminar" es el lema de la palabra "Caminar", es decir la palabra de la cual proviene. En la segunda línea "es" esta etiquetado con "VLfin" es un verbo léxico finito. En la siguientes líneas vemos que "ADJ" es la etiqueta que le asigna a los adjetivos, "PREP" a las prepocisiones, "ART" a los artículos y "NC" a los sustantivos comunes.

Luego construyo el fragmento de XML para la oración y el inicio y fin de corpus.

```
<corpus>
  <oracion>
    <palabra sint='VLinf' lema='caminar'>Caminar</palabra>
    <palabra sint='VLfin' lema='ser'>es</palabra>
    <palabra sint='ADJ' lema='bueno'>bueno</palabra>
    <palabra sint='PREP' lema='para'>para</palabra>
    <palabra sint='ART' lema='el'>la</palabra>
    <palabra sint='NC' lema='salud'>salud</palabra>
  </oracion>
</corpus>
```

Agrupamos las clases de palabras VLinf y VLfin a V, notar que este caso no tenemos varias ocurrencias de cada verbo, pero agrupar estas etiquetas es necesario ya que el sistema toma como entrada 1 solo valor de la etiqueta que representa el verbo. En caso de que no agruparamos estas etiquetas, deberíamos elegir cual de las 2 etiquetas vamos a considerar como verbo a estudiar, lo cual sería útil por ejemplo, si alguien quiere estudiar los verbos en una persona o modo particular.

```
<corpus>
  <oracion>
    <palabra sint='V' lema='caminar'>Caminar</palabra>
    <palabra sint='V' lema='ser'>es</palabra>
    <palabra sint='ADJ' lema='bueno'>bueno</palabra>
    <palabra sint='PREP' lema='para'>para</palabra>
    <palabra sint='ART' lema='el'>la</palabra>
    <palabra sint='NC' lema='salud'>salud</palabra>
  </oracion>
</corpus>
```

Ahora supongamos que vamos a considerar el verbo "ser" y la etiqueta 'sint'. También supongamos que buscamos los marcos de subcategorización verbal en el alcance 3, es decir 3 palabras a cada lado del verbo, entonces el patrón encontrado en este caso es:

[V,ADJ,PREP,ART] y en el caso que rellenara el patrón por ejemplo con la palabra NULO quedaría:

[NULO,NULO,V,ADJ,PREP,ART]

En el ejemplo si no se considerara el orden de las palabras, se tomaría el orden alfabético para las etiquetas y el patrón quedaría:

[ADJ,ART,PREP,V] ó

[ADJ,ART,NULO,NULO,PREP,V] si consideramos rellenar el patrón.

Siguiendo con nuestro ejemplo, si consideraramos incluir la marca verbal, el patrón quedaría:

[V,|,ADJ,PREP,ART]

Ahora supongamos que ejecutamos el sistema de la siguiente manera...

IRASubcat.py corpus_oracion.xml sint=V oracion lema config.txt

considerando como target tags del archivo de configuracion sint, sint-lema, que consideramos el orden de las palabras, que no incluiremos una marca verbal y que no haremos colapsado de patrones (ya que en este caso al ser solo una oración que contiene 2 verbos distintos no tenemos patrones para colapsar, y en particular poner la opción de colapsar nos haría perder el orden y la marca verbal). Y consideramos todos los verbos.

Con las opciones nombradas, obtendríamos el siguiente diccionario de subcategorización de salida:

```
<dictionary
execute="IRASubcat.py corpus_oracion.xml sint=V oracion lema config.txt"
verb_list="False" dict_exist="False" scope="3" comp_w_word="False" order="True"
tags="['sint', 'sint-lema']" lex_items="False" verbal_mark="True"
verb_min_abs_freq="0" pattern_min_abs_freq="0" colapse_patterns="False"
use_likelihood="True">
    <entry verb="caminar" count_oc_verb="1">
        <tag name="sint" different_patterns="1">
            <pattern id="|,V,ADJ,PREP" count_w_verb="1" total_count="1"
             rejected_patterns_freq_test="NO" likelihood_test="'no_paso'">
            </pattern>
        </tag>
        <tag name="sint-lema" different_patterns="1">
            <pattern id="|,V-ser,ADJ-bueno,PREP-para" count_w_verb="1" total_count="1"
             rejected_patterns_freq_test="NO" likelihood_test="'no_paso'">
            </pattern>
        </tag>
    </entry>
    <entry verb="ser" count_oc_verb="1">
        <tag name="sint" different_patterns="1">
            <pattern id="V,|,ADJ,PREP,ART" count_w_verb="1" total_count="1"
             rejected_patterns_freq_test="NO" likelihood_test="'no_paso'">
            </pattern>
        </tag>
        <tag name="sint-lema" different_patterns="1">
            <pattern id="V-caminar,|,ADJ-bueno,PREP-para,ART-el" count_w_verb="1"
             total_count="1" rejected_patterns_freq_test="NO" likelihood_test="'no_paso'">
            </pattern>
        </tag>
    </entry>
</dictionary>
```

Ahora veamos que podríamos estudiar los items léxicos en combinación con alguna característica de las palabras, ejecutemos el sistema con la misma configuración pero considerando las etiquetas: sint-lexical (recordemos que lexical era la palabra reservada para identificar la forma de las palabras). Obtenemos el siguiente diccionario de salida.

```
<dictionary
execute="IRASubcat.py corpus_oracion.xml sint=V oracion lema config.txt"
verb_list="False" dict_exist="False" scope="3" comp_w_word="False" order="True"
tags="['sint-lema']" lex_items="False" verbal_mark="True" verb_min_abs_freq="0"
pattern_min_abs_freq="0" colapse_patterns="False" use_likelihood="True">
    <entry verb="caminar" count_oc_verb="1">
        <tag name="sint-lema" different_patterns="1">
            <pattern id="|,V-ser,ADJ-bueno,PREP-para" count_w_verb="1" total_count="1"
             rejected_patterns_freq_test="NO" likelihood_test="'no_paso'">
            </pattern>
        </tag>
    </entry>
    <entry verb="ser" count_oc_verb="1">
        <tag name="sint-lema" different_patterns="1">
            <pattern id="V-caminar,|,ADJ-bueno,PREP-para,ART-el" count_w_verb="1"
```

```
            total_count="1" rejected_patterns_freq_test="NO"
            likelihood_test="'no_paso'">
           </pattern>
        </tag>
    </entry>
</dictionary>
```

Capítulo 5

Detalle de la Arquitectura Interna del Sistema

En este capítulo se presentan los detalles de los diferentes módulos del sistema. Se explica el parser de entrada y cómo se identifican los verbos, patrones y las co-ocurrencias entre ellos.

En la sección 5.4 se explicará la estrategia para identificar constituyentes opcionales (también llamados adjuntos). Después de identificar los adjuntos, se colapsan diferentes patrones que comparten los mismos argumentos y sólo difieren en cuanto a los adjuntos.

Finalmente, en la sección 5.5 se describen los diferentes filtros para establecer o descartar asociaciones entre marcos y verbos: umbral de frecuencias absolutas y relativas y test de hipótesis likelihood ratio. Estos filtros son configurados por el usuario en los parámetros de entrada.

5.1. Implementación

El sistema se implementó en Python. Python es un lenguaje de programación interpretado creado por Guido van Rossum en el año 1991. Por información histórica puede consultar la siguiente página http://es.wikipedia.org/wiki/Python.
Python se puede bajar desde la página www.python.org.

En la actualidad Python se desarrolla como un proyecto de código abierto, administrado por la Python Software Foundation.

La última versión estable del lenguaje es la 3.1.1., aunque el sistema requiere solo la 2.5

Python se utiliza como lenguaje de programación interpretado, lo que ahorra un tiempo considerable en el desarrollo del programa, pues no es necesario compilar ni enlazar.

Python permite dividir el programa en módulos reutilizables desde otros programas Python. Viene con una gran colección de módulos estándar que se pueden utilizar como base de los programas.

Python usa una arquitectura de máquina virtual, similar en concepto a máquina virtual de java. El interpretador de python compila los programas a código de máquina en mismo momento. Estos archivos compilados típicamente tienen extensión .pyc y son portables. Es decir, se puede guardar el código fuente y entregar a los usuarios finales los archivos compilados.

5.2. Parser de entrada

Para trabajar con XML algunos de los paquetes que hay disponibles para python son:

- La librería estándar de python.
- PyXML producido por Python XML Special Interest Group.
- 4suite provista por Fourthought, Inc.

La librería estándar de python provee una interfaz llamada Expat XML parser, que es la que se usó en esta tesis para el parser del corpus de entrada, y para restaurar un diccionario existente.

Luego de que el objeto parser es creado, hay funciones que devuelven elementos XML con sus atributos, con esa información se identifican las oraciones y las palabras con sus respectivos atributos.

Al comenzar la tesis se usó otra librería para el parser, la librería se llamaba libxml2. Este módulo si bien funcionaba correctamente, construía un árbol del XML dado en memoria, por lo tanto la posibilidad de ejecución del sistema dependería del tamaño del corpus y de tener los recursos de hardware suficiente para alojar el árbol en memoria, motivo por el cual no se usó en esta tesis, ya que IRASubcat pretende poder ser usado con corpus de variados tamaños y con computadoras comunes.

Con funciones de la librería Expat se creó un objeto que se encargará del parser, es decir de guardar el estado actual y se usaron otras funciones que dan elementos XML, indicando si es inicio o fin del elemento, o datos con atributos.

Para identificar los constituyentes lingüísticos de un patrón, se toma el elemento especificado por el usuario como padre de los constituyentes del patrón en una oración, y a partir de ahí se exploran todos los hijos de este nodo y se obtienen los valores de los elementos necesarios para formar el patrón.

Por ejemplo, en el siguiente elemento XML, vea figura 5.1, si el usuario especificó que "oracion" era el elemento padre a tener en cuenta, que "morfosint" la etiqueta morfosintáctica, "Vsfin" el valor de la etiqueta morfosintáctica que indica que el elemento es un verbo y que se querían tomar 1 elemento a cada lado del verbo para conformar el patrón, se obtendría el siguiente patrón: [NC ADJ]

```
<oracion>
   <palabra morfosint='ART' lema='el'>La</palabra>
   <palabra morfosint='NC' lema='casa'>casa</palabra>
   <palabra morfosint='Vsfin' lema='ser'>es</palabra>
   <palabra morfosint='ADJ' lema='lindo'>linda</palabra>
</oracion>
```

y en el sistema

Las palabras son objetos en el sistema y poseen métodos setAttribute(attr, val), y getAttribute(attr), con los cuales modificamos los atributos y obtenemos los atributos de las palabras.

Una palabra puede pertenecer a la subclase palabra-verbo.

Un objeto oración en el sistema es una lista de objetos palabra. Entre los cuales puede haber objetos palabra-verbo.

Entonces para cada oración puede haber 0 o más objetos palabra-verbo.

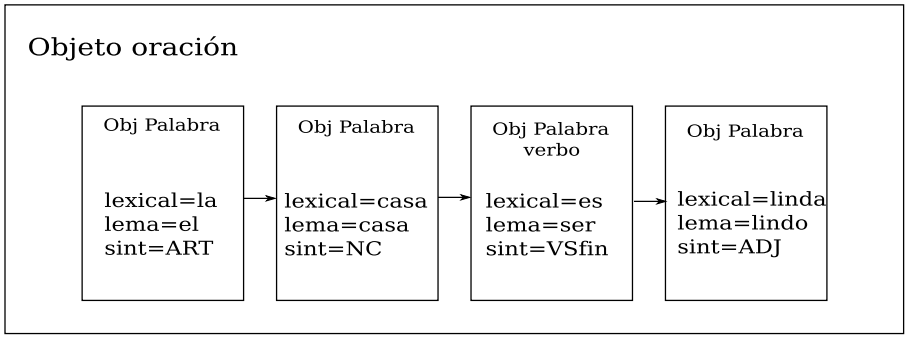

Figura 5.1: Objeto oración en el sistema.

5.3. Identificación de Patrones, Verbos y Co-ocurrencias

En la identificación de los verbos y en la identificación del patrón en la oración en el ámbito seleccionado, se tuvo en cuenta que una oración puede tener muchos verbos, y que el mismo verbo puede ocurrir varias veces en la oración.

El sistema identifica los verbos, encontrando el objeto palabra-verbo entre los objetos palabra, el cual significa que el valor del atributo de la categoría morfosintáctica indica que es un verbo.

Para identificar el patrón en cada oración para cada una de las etiquetas seleccionadas, en el ámbito seleccionado, el sistema toma una oración, busca en ella el primer verbo considerado y solicita a cada palabra (en el ámbito estudiado) el valor de cada una de las etiquetas, y así teniendo en cuenta las demás opciones (por ejemplo, agregar la marca verbal, no considerar el orden etc.), arma el patrón para cada una las etiquetas o combinación de etiquetas y con esa información el sistema actualiza un diccionario intermedio del verbo y etiqueta en cuestión, también actualiza un diccionario de totales de patrones por etiquetas, luego sigue desde la posición en la que encontró el verbo, buscando otros verbos (que hasta podría ser el mismo). Siguiendo con el ejemplo actualizamos las entradas del verbo ser y la etiqueta morfosint con el patrón [NC, ADJ]

Como ese patrón ya pertenecía al diccionario, le sumamos una ocurrencia al contador, como se muestra en la figura 5.2. En caso de que no hubiera pertenecido, agrega el patrón con contador de ocurrencias igual a 1.

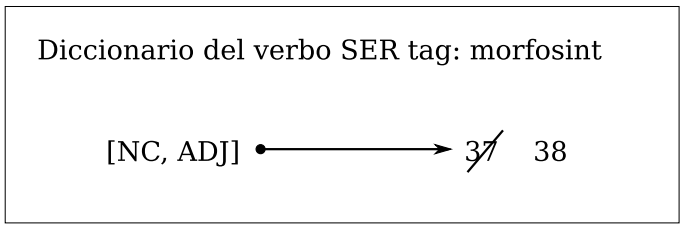

Figura 5.2: Diccionario del verbo

5.4. Detección de constituyentes opcionales

En esta sección se explica la estrategia para identificar constituyentes opcionales, que en la bibliografía lingüística se llaman adjuntos. Identificar los adjuntos es importante para identificar su complemento, los argumentos, que constituyen el núcleo del patrón y por lo tanto configuran el marco de subcategorización de un verbo. Después de identificar los adjuntos, se colapsan diferentes

patrones que comparten los mismos argumentos y sólo difieren en cuanto a los adjuntos.

Para fundamentar teóricamente la distinción entre argumentos y adjuntos se consultó material de (Dowty 1979), (Fillmore 1968) y (Hale and Keyser 2001).

Un argumento verbal es un complemento o un constituyente requerido obligatoriamente por un cierto verbo para formar una oración gramatical.

Cuando un constituyente es requerido obligatoriamente por un verbo se dice que el verbo "subcategoriza" o "rige" dicho complemento o argumento. Los complementos del verbo no obligatorios se califican de adjuntos u opcionales.

Por lo tanto se dice que los argumentos son centrales en la configuración del verbo y los adjuntos son opcionales.

Por ejemplo en español todos los verbos, excepto los meteorológicos (llueve, nieva, ...), requieren un sujeto obligatoriamente, los verbos transitivos requieren al menos otro argumento más denominado complemento directo, y algunos verbos pueden requerir un sintagma preposicional como por ejemplo: "Pienso en ti".

La elisión de argumentos es otra cosa a tener en cuenta. La elisión de argumentos es la omisión de alguna entidad lógica necesaria para el sentido de la frase. Estos elementos se pueden inferir en el contexto de la oración.

Es común que los hablantes eliminen vocablos o frases en su discurso, siempre que el receptor pueda inferir aquellos elementos del contexto de la oración.
Ejemplo:

1. Los animales pastaban en el monte. Tenían hambre y sed.

2. Lo bueno, si breve, dos veces bueno. (Gracián)

En la segunda cláusula de (1) "se ha eliminado" el sujeto. Es decir, de interpretación lógica de la frase '[Los animales] tenían hambre y sed', el elemento entre corchetes que hace la función de sujeto no aparece en la segunda oración de (1).

En la segunda cláusula de (2) "se ha eliminado" el verbo ser, es decir, la frase sería "Lo bueno, si [es] breve, dos veces bueno."

Por lo tanto podemos decir que algunos casos no es que no sea obligatorio un componente sino que está elidido (el componente está, pero no se ve explícitamente).
En general podemos ver cuando estudiamos un verbo particular que algunos patrones co-ocurren con el verbo mucho menos (significativamente menos) que otros. Como consecuencia de esto, tienden a desaparecer cuando aplicamos un simple test de hipótesis, como por ejemplo el test de mínima frecuencia relativa para considerar la co-ocurrencia.

La detección de constituyentes opcionales es una tarea útil en nuestro caso ya que permite atacar el problema de escasez de datos (data sparseness), y es descriptivamente adecuado por la dicotomía argumento-adjunto. Cuando distinguimos argumentos de adjuntos, consideramos que el patrón se caracteriza únicamente por los argumentos. Así, se reduce el problema de escasez de datos porque se introduce abstracción, ya que se pasa a considerar como el mismo a varios patrones que se consideraban distintos. De esta forma reducimos la dimensionalidad y la dispersión de los datos.

Para reducir el data sparseness en un corpus también se pueden agrupar clases de palabras. Por ejemplo si en el corpus hubiera palabras que corresponden a Sustantivos, y palabras que co-

rresponden a Nombres Propios, podríamos agrupar ambas clases de palabras.

Métodos para detectar constituyentes opcionales

Vamos a describir varios métodos para identificar constituyentes opcionales en los patrones asociados a un verbo, y vamos a detallar el método que finalmente implementamos en la tesis.

Un posible método consiste en recorrer los patrones que co-ocurren con un verbo dado, y a los constituyentes que ocurran en todos los patrones (o en una mayoría significativa), llamarlos "argumentos", y, a los demás, adjuntos.

Otro método para colapsar patrones es construir una Trie con los constituyentes de los patrones. Una Trie (o *prefix tree*) es una estructura de datos de árbol ordenado que se usa para almacenar un array asociativo, donde las claves son usualmente strings. La posición en el árbol muestra cuál es la clave del elemento, todos los descendientes de un nodo tienen un prefijo común del string asociado con aquel nodo y la raíz tiene el string vacío.
Los nodos suelen almacenar valores.
Entre las ventajas de usar una Trie podemos nombrar:

- Búsqueda de claves más rápida. La búsqueda de una clave de longitud m tendrá en el peor de los casos un coste de O(m). (En nuestro caso m será la cantidad de constituyentes que tenga el patrón).
- Menos espacio requerido para almacenar gran cantidad de cadenas pequeñas, puesto que las claves no se almacenan explícitamente.
- Mejor funcionamiento para el algoritmo de búsqueda del prefijo más largo.

Mediante esta estructura de datos se pueden representar los patrones, ordenando los constituyentes de cada patrón por su frecuencia de ocurrencia con el verbo, de forma que los constituyentes que ocurren más veces con el verbo queden más cerca de la raíz de la trie. Este ordenamiento trata de capturar la intuición de que los argumentos co-ocurren con el verbo más frecuentemente que los adjuntos.

Bajo esta asunción y en esta representación, los argumentos quedarán sistemáticamente más cerca de la raíz de la trie, mientras que los adjuntos se situarán hacia las hojas. Las características que indican que un elemento puede ser adjunto en una trie son:

- Que ocurra en una hoja
- Que haya una diferencia significativa entre el número de ocurrencias del padre y el hijo

La ordenación de los elementos de la trie por cantidad de ocurrencias, es ya muy indicativa de la argumentalidad de un elemento, y en realidad lo único que nos queda determinar es qué punto de una rama cortar para diferenciar entre argumentos y adjuntos.

Hay diferentes posibilidades para identificar los adjuntos:

- Ver si los elementos finales de un patrón ocurren menos o igual que su padre (considerando como padre al patrón que contiene todos los constituyentes del patrón considerado menos el último, que es el que se está evaluando como adjunto).
  - Se puede tomar el valor absoluto,
  - o una proporción (Este caso sería muy adecuado si el patrón contiene los argumentos y 1 solo adjunto).
- Ver si un mismo elemento que ocurre como hoja no ocurre en otras posiciones.
  - Opción 1: si ocurre en otras posiciones, no es adjunto.

    - Opción 2: si no ocurre en otras posiciones, se chequea que cumpla la condición en el 80 % de los casos para declararlo adjunto.

- Ver si la diferencia de cantidad de ocurrencias entre el padre y el hijo es significativa (tomando como padre a los subpatrones del hijo), notar que aquí estaría tomando la trie sin elemento de fin de patrón), y ahí cortar hasta el final de la rama.

Además, es importante que la detección de adjuntos conserve **coherencia**: si un elemento se detectó como adjunto en un verbo, todos los patrones del verbo que posean ese elemento lo tendrán como adjunto.

En esta tesis se implementó la detección de constituyentes opcionales según la opción 2 nombrada anteriormente y respetando la coherencia de la siguiente manera:

- Se ordenaron los patrones según el ranking de los constituyentes de mayor a menor número de ocurrencias (eliminando marca verbal y símbolos de relleno).

- Se recorrió el diccionario de verbos-patrones para cada verbo a fin de encontrar constituyentes que ocurrían sólo en las hojas.

    - Para cada constituyente que ocurrió solo en hojas, se buscó en la trie la cantidad de ocurrencias del padre, y se la comparó con la cantidad de ocurrencias del hijo. Si para el 80 % de los patrones se cumplía que el contador del padre era mayor o igual que el contador del hijo, se lo consideraba adjunto, se colapsaban los patrones eliminando los adjuntos y uniendo los argumentos, y se creaba una nueva trie con los nuevos patrones. Si no, se lo consideraba argumento en la trie actual, lo que no implica que no pueda considerarse adjunto en las subsiguientes iteraciones, y se pasaba a la siguiente iteración.

Notar que al tomar que se cumpla para el 80 % y reemplazarlo en todas las ocurrencias, cumplimos con la coherencia.

Para ilustrar la forma de colapsar los patrones veremos un ejemplo.
Supongamos que tenemos las siguientes 6 oraciones para el verbo tener

1. La casa tiene un lindo patio con pileta
2. La casa tiene un patio chico
3. La casa tiene patio
4. La casa tiene un gran patio con vista al mar
5. Los departamentos tienen patios internos
6. Las casas tienen grandes patios para descanzar

y supongamos que el resultado de un analizador sintáctico que provee información de SN (sujeto nominal), OD (objeto directo), y PP-prep (frase preposicional) nos dá los patrones respectivos de las oraciones anteriores:

```
[SN OD PP-con], [SN OD], [SN OD], [SN OD PP-con],[SN OD] y [SN OD PP-para]
```

Buscamos las cantidades de los constituyentes del verbo y tag en cuestión a partir de los patrones encontrados a fin de ordenar los patrones:

```
SN --> 6
OD --> 6
PP-con --> 2
PP-para -->1
```

Ordenamos por cantidad de ocurrencias de mayor a menor (notar que los constituyentes que tienen igual contador pueden ser ordenados de cualquier modo, pero con el mismo orden para todos los patrones)

```
SN OD PP-con PP-para
```

Ahora ordenamos los patrones según el orden dado anteriormente (en este caso es el mismo orden).

```
[SN OD PP-con]
[SN OD]
[SN OD]
[SN OD PP-con]
[SN OD]
[SN OD PP-para]
```

Agregamos estos patrones a la trie como se muestra en la figura 5.3:

```
[SN OD] ->3
[SN OD PP-con] ->2
[SN OD PP-para] ->1
```

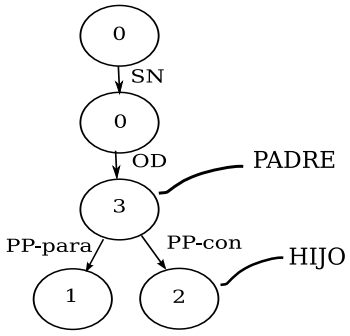

Figura 5.3: Trie inicial

Ahora recorremos todos los constituyentes que son hojas de los patrones (del diccionario, no de la trie), en este caso son hojas "OD","PP-con" y "PP-para"
Como se dá que OD está en otra posición que no es hoja, lo descarto por esta iteración (la primera).

El patrón en el que se cumple que el constituyente PP-con es hoja es [SN OD PP-con], con 2 ocurrencias. Al buscar el número de ocurrencias del patrón padre del patrón considerado, si éste no existiera, nos daría el valor 0. En este caso, existe y nos dá el valor 3, como 3 es mayor que 2, y era el único patrón que tenía como hoja a PP-con, el 100 % de ocurrencias de PP-con cumple la condición. Entonces aceptamos que PP-con es un adjunto opcional, y recortamos el patrón que lo contiene, quedando:

```
[SN OD]{PP-con:2} -> 5
```

Continuamos con [SN OD PP-para] que es el único patrón que tiene PP-para como hoja, buscamos en la trie el contador del padre y nos dá que el contador del padre es mayor que el del patrón y como era la única ocurrencia de PP-para, el 100 % de casos cumplen la condición, por lo tanto el constituyente PP-para es considerado adjunto. El patrón que queda es:

```
 [SN OD]{PP-con:2, PP-para:1} -> 6
```

como se puede ver en la figura 5.4, y pasamos a la siguiente iteración.

Recorremos las hojas, la única hoja que hay es "OD". Al buscar en la trie el valor del padre, vemos que el padre no esta, lo cual nos devuelve el valor 0. Como no se cumple que la cantidad de ocurrencias del hijo sea menor o igual que la del padre, lo consideramos argumento, y pasamos a la siguiente iteración.
Recorremos las hojas, la única hoja es "OD", y como en la anterior también lo era, llegamos al FIN del algoritmo.

Notar que en este ejemplo realizamos todas las iteraciones que pudimos, pero si el usuario hubiera seleccionado un número máximo de iteraciones en el archivo de configuración, hubieramos hecho sólo ese número de iteraciones.

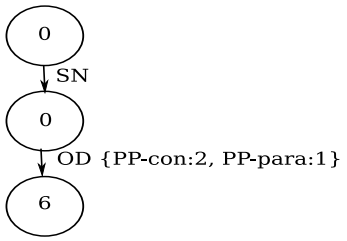

Figura 5.4: Trie final con 2 constituyentes opcionales

Este proceso se realiza antes de filtrar patrones por frecuencia mínima, ya que el colapsador podría agrupar patrones que el test de frecuencia mínima hubiera filtrado, pero el test de frecuencia de verbos se aplica antes del colapsador, ya que no es necesario colapsar patrones de verbos que no estarán en el diccionario de salida.

Notar que el método de detección de constituyentes opcionales (adjuntos) podría detectar adjuntos incluso si el patrón nunca ocurrió con argumentos solamente, sino siempre con adjuntos, en estos casos detectaría sólo algunos adjuntos. En particular no detectaría el adjunto más próximo a los argumentos.

5.5. Identificación del Marco de subcategorización

El marco de subcategorización es el conjunto de patrones que admite el verbo en oraciones gramaticales en un idioma particular.

En el marco de subcategorización se pueden diferenciar o no los adjuntos.

Lo que genera IRASubcat es una aproximación al marco de subcategorización, ya que obtiene los patrones de realización de un verbo en un corpus particular, lo cual puede mostrar grandes diferencias con una muestra más significativa de la lengua. Es de esperarse que a medida que aumentamos el tamaño del corpus estudiado las diferencias entre el marco encontrado y el marco real sean mucho menores.

A fin de filtrar errores y comportamientos no significativos de la información verdaderamente relevante se realizan diversos test de hipótesis. Éstos nos ayudarán a admitir el patrón obtenido en el corpus en el marco de subcategorización del verbo en el idioma es cuestión, con un cierto grado de confiabilidad, o a rechazarlo.

Para seleccionar los tests implementados en IRASubcat, se estudiaron los test de hipótesis aplicados a tareas similares, entre ellos podemos nombrar el test binomial, el test de t, el likelihood

ratio y el test de frecuencias.

Para decidir qué test de hipótesis aplicar tenemos que tener en cuenta que es lo que asume el test. Esta información fue obtenida de (Dunning 1993).

En procesamiento de lenguaje natural también se suele usar información mútua (pointwise mutual information), con la fórmula:

$$SI(x, y) = \log(p(x, y)/p(x) * p(y)) \tag{5.1}$$

que es simétrica, ya que no tiene en cuenta el orden de ocurrencia de los elementos lingüísticos involucrados.

Esta medida no modela bien los elementos con frecuencias bajas, que suelen resultar en valores muy altos de IM que son irreales. Y como dijimos anteriormente las frecuencias bajas son comunes en PLN. Por lo tanto no se usó esta medida.

El log-likelihood ratio modela mejor los elementos que ocurren pocas veces porque asume una distribución de probabilidad más adecuada al fenómeno real, en concreto la distribución χ^2.

En esta tesis se implementó el test log-likelihood ratio, y se deja como trabajo futuro la implementación del test binomial.

En los ejemplos estudiados se vió que la frecuencia relativa funciona mejor que cualquier test. IRASubcat también provee la funcionalidad de filtrar patrones y verbos mediante un test de frecuencia.

5.5.1. Test de Hipótesis Likelihood Ratio

Cuando aplicamos test de hipótesis para adquirir marcos de subcategorización verbales, la tarea es examinar cuando hay evidencia suficiente que avale la genuina asociación entre un verbo particular 'V_i' y el marco particular 'M_j'.

Se formula una hipótesis nula (H_0) la cual es asumida verdadera si no hay evidencia de lo contrario. Si hay evidencia de lo contrario se rechaza la hipótesis nula H_0 y se acepta la hipótesis alternativa (H_1).

Cada ocurrencia del verbo V_i tiene probabilidad mayor que 0 de aparecer con el marco M_j, incluso si este verbo no aparece con ese marco, en cuyo caso se le dá una probabilidad muy chiquita. La hipótesis nula del test es:
H_0: no hay asociación entre el verbo V_i y el marco de subcategorización M_j y la hipótesis alternativa H_1 es que hay tal asociación.

Este test es un test de una sola cola ya que en H_1 la dirección de la asociación es una correlación positiva entre el verbo V_i y el marco M_j.

La probabilidad esperada de que M_j ocurra con el verbo V_i si H_0 sea verdadera (este es el valor de la χ^2 para la confiabilidad dada) es comparada con la probabilidad observada de co-ocurrencia obtenida desde el corpus.

Si la probabilidad observada es más grande que la probabilidad esperada, rechazamos H_0 y aceptamos H_1, y si no, aceptamos H_0.

El estadístico -2 $\log \lambda$ es calculado como sigue:

$$log - likelihood = 2[logL(p1, k1, n1) + logL(p2, k2, n2) - logL(p, k1, n1) + logL(p, k2, n2)] \quad (5.2)$$

donde:

$$logL(p, k, n) = k * \log p + (n - k) * \log(1 - p) \quad (5.3)$$

y

$$p1 = \frac{k1}{n1}$$
$$p2 = \frac{k2}{n2}$$
$$p = \frac{k1 + k2}{n1 + n2}$$

n1 = cantidad de ocurrencias del verbo considerado
n2 = cantidad de ocurrencias de otros verbos
k1 = cantidad de ocurrencias del marco con el verbo considerado
k2 = cantidad de ocurrencias del marco con otro verbo

Notar que p1 es la probabilidad de ocurrencia del marco dado el verbo, y p2 es la probabilidad de ocurrencia del marco dado que no se dió el verbo y p es la probabilidad del marco dado cualquier verbo.

Estamos interesados en los casos donde p1>p2, para estos casos, el valor de -2log λ es comparado con el valor obtenido de la tabla χ^2 para los 4 umbrales de confiabilidad considerados por el sistema (0.90, 0,95, 0.99, 0.995), si el valor de -2log λ es mayor que la probabilidad esperada en el umbral 0.995, rechazamos la H_o con nivel de confianza 99.5 %. Si no, evaluamos con el nivel de confianza de 0.99 y así hasta decidir si pasó algún nivel. Si no pasó el test de hipótesis en ningún nivel de confianza, el sistema lo clasifica en la clase "no pasó".
En los casos que no se cumpla la condición p1>p2, el sistema clasifica el caso como 'NO_DECIDE'.

5.5.2. Umbral de Frecuencia Absoluta para la cantidad de ocurrencias del verbo

Este es un filtro simple que es muy útil ya que, si tenemos suficientes ejemplos de oraciones con un verbo, tenemos más probabilidad de haber acumulado muchas ocurrencias del mismo patrón, lo que significaría que ese patrón tiene chance de ser considerado marco de subcategorización para el verbo considerado.

En contraste a lo anterior cuando tenemos pocas, por ejemplo 1 oración con un verbo a estudiar, no podremos decir mucho de ese verbo, por lo menos con el test de hipótesis aplicado en esta tesis; por lo tanto, y para no considerar asociaciones que no acumulen suficiente evidencia, el usuario podría cambiar este valor, en el archivo de configuración.

5.5.3. Umbral de Frecuencia Relativa para la ocurrencia del patrón

También podríamos querer considerar solo aquellos patrones que ocurren una cantidad mínima de veces con el verbo y así filtrar los patrones con pocos ejemplos con el verbo.

Para filtrar con este test, basta seleccionar la mínima frecuencia relativa en el archivo de configuración.

Parte II

Aplicación

Capítulo 6

Aplicación a un corpus del Español

En este capítulo se presenta una aplicación de IRASubcat al procesamiento de un corpus en español con un intensivo análisis lingüístico previo manual. Se describe el corpus y los diferentes análisis asociados al mismo, la forma de ejecutar el sistema, los experimentos llevados a cabo y se dá un análisis de los resultados obtenidos.

6.1. Datos del corpus

El corpus fuente con el que se trabaja es el corpus SenSem (Fernández 2004). Está formado por unos 13 millones de palabras extraídas de las versiones online de un diario escrito en español (El Periódico).

De este corpus se han seleccionado aleatoriamente 25.000 frases, 100 para cada uno de los 250 verbos más frecuentes del español actual.

Cada oración se etiqueta de acuerdo con el sentido verbal que ejemplifica, el tipo de complementos (argumentos determinados por el verbo o adjuntos opcionales), así como la categoría (sintagma nominal, sintagma verbal, sintagma preposicional, etc.) y la función sintáctica (sujeto, objeto directo, etc.) y semántica (agente, destinatario, tema) de estos.

También se incluye el tipo de semántica oracional que expresa la oración tanto en relación con la información aspectual como por lo que se refiere a la construcción.

6.2. Preprocesamiento del corpus

El corpus no estaba en XML válido, por lo tanto se implementó un preproceso que tomaba como entrada el corpus y devolvía el mismo corpus en XML válido. (No estaban especificados los finales de los elementos, y lema_verbo aparecía en el ámbito 's' y el sistema necesitaba que apareciera en las 'phr', por lo tanto se agregó esta información.)

Este corpus provee muchas etiquetas asociadas a propiedades lingüísticas. En particular en ésta tesis se analizaron "cat" y "fs", la primera refiere a la categoría morfológica de los sintagmas y la segunda a su función sintáctica.

Aquí se mostrarán los resultados explorando únicamente patrones caracterizados por la función sintáctica de sus constituyentes.

Se denomina función sintáctica al servicio que una palabra desempeña a otra u otras o, de forma más precisa, a las relaciones de combinación o relaciones sintagmáticas que una palabra

mantiene con las demás palabras de su contexto.

Las funciones sintácticas más simples son las de sujeto y predicado. El sujeto es el tema o asunto o soporte de que se habla; el predicado es lo que se dice o predica o comenta o aporta sobre ese sujeto:

Por ejemplo:
Pedro (función sintáctica sujeto) come (función sintáctica predicado).

Las funciones sintácticas están desempeñadas en el lenguaje por distintas clases de palabras (el sustantivo Pedro en el caso del sujeto anterior, el verbo come en el caso del predicado anterior).

Las funciones sintácticas ligan las palabras entre sí en el eje sintagmático o combinatorio de la expresión, frente al eje paradigmático o sustitutorio de la expresión, y muchas veces estan marcadas por determinados morfemas que sirven para establecer relaciones de concordancia.

6.3. Configuración de IRASUBCAT

Para realizar los experimentos, se ejecutó IRASubcat con la siguiente configuración:

TO CONSIDER VERB LIST = NO
No se consideró un subconjunto predeterminado de verbos.

DICTIONARY EXIST = NO
No se actualizó ningún lexicón existente.

LONG OF SIDE THE VERB FOR PATTERN = ALL
Se consideró como ámbito a toda la oración.

COMPLETE WITH WORD = NO
No se completó el patrón con ninguna palabra especial.

ORDER OF TAGS = NO
No se tuvo en cuenta el orden de las etiquetas encontradas para armar el patrón.

TARGET TAGS = fs
Se consideró la función sintáctica de los constituyentes para armar el patrón.

USE LEXICAL ITEMS = NO
No se usó la forma de la palabra como constituyente del patrón.

INTRODUCE VERBAL MARK = NO
No se introdujo marca verbal.

COLAPSE PATTERNS = YES
Se colapsaron los patrones que contenían constituyentes detectados como adjuntos por el sistema.

MAX ITERATION FOR FIND COLAPSE PATTERNS = FALSE
Se realizaron todas las iteraciones que se pudieron para colapsar patrones.

MINIMAL ABSOLUTE FREQUENCY = 0
No se filtró por cantidad mínima de verbos.

MINIMAL RELATIVE FREQUENCY TO CONSIDER PATTERN = 0
No se filtraron patrones por tener pocas ocurrencias con el verbo.

USE LIKELIHOOD RATIO TEST = YES
Se usó test de hipótesis likelihood ratio.

6.4. Descripción de los experimentos

Para el español se estudiaron 20 sentidos verbales que fueron elegidos por ser los que mayores cantidades de ejemplos tenían en el corpus.

Los sentidos estudiados fueron: necesitar_1, lograr_1, escribir_1, tardar_1, anunciar_1, desear_1, afirmar_2, casar_1, merecer_1, negociar_1, votar_1, descartar_1, detectar_1, intentar_1, llenar_3, decidir_1, financiar_1, controlar_1, efectuar_1 y conseguir_1.

Se compararon los patrones asociados a cada verbo en la base de datos de SenSem para la función sintáctica, con los 10 patrones más frecuentes encontrados por el sistema para cada sentido, para la función sintáctica.

En esta comparación había pequeñas discrepancias de forma, ya que en los patrones identificados por IRASubcat se colapsaban patrones con los mismos argumentos, abstrayéndose de los adjuntos, y no se conservaba el orden de los constituyentes. En cambio, en la base de datos de SenSem, los patrones conservan orden y no se descartan los adjuntos.

Para solucionar estas discrepancias y poder realizar la comparación, se estableció una correspondencia entre cualquier par de patrones de IRASubcat y SenSem siempre que:

- todos los constituyentes identificados como argumentos en el patrón de IRASubcat estuvieran presentes en el patrón de SenSem,
- todos los constituyentes del patrón de SenSem se encontraran en el patrón de IRASubcat, independientemente de su estatus como adjuntos o como argumentos y del orden en los que se encontraran.

Usaremos las medidas de precisión y cobertura para evaluar el rendimiento del sistema con los distintos umbrales del test de hipótesis likelihood ratio.

La **Precisión** puede ser vista como una medida de exactitud o fidelidad, mientras que la **Cobertura** (en inglés *recall*) sería una medida de completitud.

En el área de recuperación de información la **Precisión** es definida como el número de documentos (en nuestro caso, patrones) relevantes recuperados en una búsqueda dividido por el número total de documentos recuperados, vea ecuación 6.1. La **Cobertura** es definida como el número de documentos relevantes recuperados sobre el total de documentos relevantes existentes (los cuales deberían ser recuperados), vea ecuación 6.2.

$$Precision = \frac{\#Patrones\ Relevantes \cap \#Patrones\ Recuperados}{\#Patrones\ Recuperados} \tag{6.1}$$

$$Cobertura = \frac{\#Patrones\ Relevantes \cap \#Patrones\ Recuperados}{\#Patrones\ Relevantes} \tag{6.2}$$

filtro aplicado	Precisión	Cobertura	medida-F
frecuencia	.79	.70	**.74**
likelihood ratio 90 %	.42	.46	.39
likelihood ratio 95 %	.38	.42	.32
likelihood ratio 99 %	.31	.36	.22
likelihood ratio 99.5 %	.25	.28	.14

Cuadro 6.1: Resultados para diferentes filtros para asociar patrones con verbos en IRASubcat: teniendo en cuenta todos los patrones más frecuentes, o bien aplicando test de hipótesis.

Una medida que combina la precisión y la cobertura es su media armónica, también llamada **medida F** (*F-measure* o *F-score*), vea la ecuación 6.3.

$$medida - F = 2 * \frac{Precision * Cobertura}{Precision + Cobertura} \tag{6.3}$$

Dadas las diferencias entre los patrones obtenidos por IRASubcat y los que se encuentran en la base de datos de SenSem, realizamos unas adaptaciones al cálculo de precisión y cobertura:

$$Precision' = \frac{\#Patrones - IRA\ con\ correspondencia\ en\ Patrones - SenSem}{\#Patrones - IRA} \tag{6.4}$$

$$Cobertura' = \frac{\#Patrones - SenSem\ reconocidos\ en\ Patrones - IRA}{\#Patrones - Sensem} \tag{6.5}$$

donde
$Patrones - SenSem$ = patrones del sentido verbal en la Base de Datos de SenSem
$Patrones - IRA$ = 10 patrones más frecuentes identificados por IRASubcat para el sentido verbal
$Patrones - IRA\ con\ correspondencia\ en\ Patrones - SenSem = Patrones - IRA$ que, con las alteraciones de orden e incorporando adjuntos permitidos, pueden ser identificados con un $Patron - Sensem$
$Patrones - SenSem\ reconocidos\ en\ Patrones - IRA = Patrones - SenSem$ que, con las alteraciones de orden y eliminando adjuntos permitidos, pueden ser identificados con un $Patron - IRA$

Los adjuntos permitidos son aquellos que son reconocidos como adjuntos por IRASubcat.

6.5. Análisis de Resultados

Recordemos que se estudió el funcionamiento de IRASubcat por comparación con la información codificada en SenSem para 20 verbos. Se compararon los 10 patrones más frecuentes asociados a cada verbo con los patrones asociados al verbo en la base de datos oracional SenSem. Los patrones se obtuvieron a partir de la característica de función sintáctica de los constituyentes oracionales.

En total se estudiaron 185 patrones de IRASubcat, ya que para algunos verbos se encontraron menos de 10 patrones. Los verbos estudiados tenían 445 patrones en SenSem.

Los resultados de estos pequeños experimentos muestran cómo el test de hipótesis filtra muchos patrones que son correctos, y por lo tanto aplicarlo lleva a una reducción de la precisión y la cobertura. En cambio, el filtro de frecuencias presenta un comportamiento mucho mejor, como se puede ver en la Tabla 6.1. Efectivamente, los resultados para los patrones más frecuentemente asociados con un verbo sin aplicar filtro de test de hipótesis, que se pueden ver en la primera fila, son mucho mejores que los resultados aplicando el test de hipótesis en cualquier nivel de confianza.

Como se puede observar en la tabla y como era de esperarse, a medida que aumentamos el umbral de confiabilidad del test de hipótesis disminuye el número de patrones aceptados y por lo tanto disminuyen también las medidas de precisión y cobertura.

Como trabajo futuro queremos implementar otros tests de hipótesis y observar si el comportamiento de los tests mejora con mayor cantidad de datos.

Capítulo 7

Aplicación a un corpus del Ruso

En este capítulo se presenta una aplicación de IRASubcat al procesamiento de un corpus en ruso sin ningún análisis lingüístico previo. Se comentarán las características del corpus en ruso y el procesamiento previo a la ejecución del sistema con dicho corpus. Después, se describirán los experimentos llevados a cabo y se analizarán los resultados obtenidos.

7.1. Datos del Corpus

El corpus ruso es un corpus plano (en cuanto a etiquetado de palabras) codificadas en cirílico de 1.000.000 de palabras. Está compuesto por 360 archivos, con artículos provenientes de "Biblioteca Ogonka" desde el año 1976 hasta el año 1990, artículos provenientes de "Gonyok" desde el año 1996 hasta el año 2001 y de "Pravda" desde el año 2000 hasta el año 2002.

Cada uno de los archivos mencionados tiene etiquetas que marcan el comienzo y fin de las frases. Por este motivo en este caso se estudiaron las frases. Dicho corpus fue provisto por gentileza de Toni Oliver, recopilado para su trabajo de tesis doctoral (Gonzalez 2004).

7.2. Preprocesamiento Lingüístico

Como el corpus no proveía ningún etiquetado para las palabras lo procesamos con el (Schmid 1994), un etiquetador morfosintáctico, si bien lo hemos nombrado a lo largo de la tesis, en la siguiente subsección se darán las características específicas de dicho analizador.

7.2.1. Descripción del analizador morfosintáctico

Un tagger es un etiquetador y desambiguador morfosintáctico, que asigna una categoría morfosintáctica a las palabras de un texto, desambiguando cuando es necesario.
El TreeTagger es un tagger probabilístico que usa árboles de decisión para asignar la categoría morfosintáctica a las palabras y desambiguarlas.

Veamos un ejemplo: "El hombre bajo toca el bajo bajo la escalera".

En éste caso la palabra "bajo" aparece en la oración tres veces, pero en cada una de las tres veces aparece en distinto rol: primero es un adjetivo, después un sustantivo, y por último una preposición.

Básicamente el TreeTagger obtiene el árbol de decisión a partir de un conjunto de trigramas de entrenamiento.

Para el inglés se tomaron 2.000.000 de palabras para entrenamiento y 100.000 palabras para testing del Penn-Treebak obteniendo 96,36 % de exactitud.

TreeTagger es un analizador morfológico que también dá información sobre lema, y es independiente de lengua.

La herramienta aprende un analizador para cada lengua para la que se le provee un corpus de entrenamiento.

Para el idioma ruso, existe una versión de TreeTagger que ha sido entrenado para encontrar la siguiente información para armar las etiquetas:

```
MSD    tag
Animate    si es un objeto animado (con vida) o no
Aspect    perfectivo/imperfectivo
CATEGORY    nombre, verbo, adverbio, adjetivo, etc.
Case    nominativo, acusativo, genitivo, dativo, instrumental, locativo o preposicional
Definiteness    definido/indefinido
Degree    neutro/comparativo/superlativo
Gender    masc/fem/neutro
Number    sing/pl
Person    1a, 2a, 3a
Tense    presente, pasado, futuro
VForm    modo: gerundio, participio, indicativo, etc.
Voice    activa, pasiva, media
```

7.2.2. Etiquetado del corpus en ruso

El corpus ruso estaba dividido en frases, con etiquetas de inicio y fin de frase, pero no contenía ningún etiquetado en las palabras, por lo tanto lo enriquecimos con el análisis morfosintáctico del Tree-Tagger y así armamos el archivo XML de entrada al sistema.

Además, hicimos algunos agrupamientos de clases de palabras para obtener una mayor abstracción y así minimizar el problema de escasez de datos (*data sparseness*). En primer lugar, agrupamos los numerales los sustantivos. También reemplazamos las etiquetas de todos los tipos de verbos, que proveían mucha información morfológica (persona, número, tiempo, modo, etc.), por "V". En las etiquetas de los pronombres, sustantivos y adjetivos dejamos sólo el caso, ignorando información de género, número, etc.

Por lo tanto, las etiquetas que quedaron en el corpus son:
Ax, siendo A que indica un adjetivo y 'x' uno de los posibles casos nombrados anteriormente.
C, una conjunción, como por ejemplo "и" y "что" ("y" y "que").
I, interjección.
Nx, indica un sustantivo, y 'x' el caso, recordar que a los numerales (M) los habíamos agrupado con los sustantivos.
Px, es un pronombre y 'x' alguno de los casos.
Q, denota una partícula.
R, indica un adverbio.
Sx, es una adposición y 'x' alguno de los casos.
V, denota un verbo. Notar que por más que teníamos información del caso para el verbo, agrupamos esas clases de palabras.

En ruso las oraciones se rigen y se construyen según 6 casos: **nominativo**, **genitivo**, **dativo**, **acusativo**, **instrumental** y **locativo** o **preposicional**.

El caso **nominativo** (también llamado primer caso) es un caso que se aplica a sintagmas nominales en función de sujeto. La forma canónica de la palabra también se expresa en nominativo.

El **genitivo** (también llamado segundo caso) es un caso de los sustantivos que indica que un nombre es un complemento nominal de otro. Normalmente la relación "(el) X de Y" se expresa en las lenguas que tienen caso genitivo como "(el) X Y*", donde el asterisco * indica que la palabra que representa Y tiene una terminación especial o lleva un morfema específico que indica que tiene caso genitivo.

El genitivo es también denominado como posesivo, aunque no hay que confundirlo con el caso posesivo, debido a su corriente uso para denotar esa relación. Sin embargo, esta segunda denominación se queda corta ya que el genitivo no solo cubre relaciones de posesión, sino también muchas otras tales como 'material del que está' hecho, objeto relacionado con,..., etc'.

El caso **dativo** (también llamado tercer caso) se aplica a sustantivos y pronombres. Este caso marca normalmente el complemento indirecto, por lo que sirve para expresar la persona o cosa que recibe el daño o provecho de la acción verbal. Así que responde a las preguntas: a quién? o para quién?, formuladas al verbo. Por ejemplo, en la oración: "el niño escribe una carta *a su padre*", el dativo sería "a su padre". Pero además existen otros usos como el de posesión, como por ejemplo en latin vulgar y, en menor medida, el latín clásico.

En el caso **acusativo** (también llamado cuarto caso) se suele ver la expresión de una relación inmediata entre el verbo y el objeto al que se refiere la acción verbal, siendo, por tanto, el caso por excelencia del complemento directo.

En lingüística, el caso **instrumental** indica el instrumento por medio del cual el sujeto realiza una acción. El instrumento bien puede ser un objeto físico o bien un concepto abstracto.

El caso **prepositivo** (o preposicional) es el término que se emplea en gramática del ruso para designar el caso locativo. Este nombre excepcional se debe a que sólo ocurre en esta lengua como complemento de una preposición. Este caso se corresponde, con ciertas reservas, con las expresiones preposicionales 'en' o 'al lado de', e indica la localización final de una acción o del momento de la acción.

Resumiendo, los casos que encontramos en ruso son los siguientes:

1. **Nominativo** (actor): responde a quién es quién?, que es que?
2. **Genitivo** (dependencia): responde a de quién?, de que?, de dónde?
3. **Dativo** (atribución): a quién?, a que?
4. **Acusativo** (objeto): quién?, que?, dónde?
5. **Instrumental** (medio): con quién?, con que?, con ayuda de que?
6. **Locativo** (localización): dónde?, en que lugar? (sin movimiento)

Esta rica variedad de casos va a hacer posible que podamos inferir la función sintáctica que está ejerciendo un determinado sintagma nominal a partir de su forma. Es decir, vamos a poder inferir los patrones sintácticos de los verbos a partir de la información que nos dé el analizador morfolsintáctico.

7.3. Configuración de IRASubcat

Para estos experimentos, ejecutamos el sistema con el siguiente archivo de configuración:

TO CONSIDER VERB LIST = 3verbos.txt

3verbos.txt es un archivo en el que guardé los verbos a estudiar.

DICTIONARY EXIST = NO
No se actualizó ningún diccionario existente.

LONG OF SIDE THE VERB FOR PATTERN = 2
Tomé 2 como un número de etiquetas que se buscan alrededor del verbo, es decir, el alcance del patrón.

COMPLETE WITH WORD = NO
En este caso se decidió no completar los patrones con ninguna palabra, para poder hacer luego un análisis de argumento-adjunto, es decir colapsar patrones con sub-patrones.

ORDER OF TAGS = NO
En el idioma ruso el orden de las palabras no es muy importante, una frase se puede decir de distintas formas, por lo tanto ya que el idioma no es tan estricto con respecto al orden de las palabras se decidió no tener en cuenta el orden.

TARGET TAGS = sint
Se consideró la informacion morfosintáctica proporcionada por el TreeTagger, que se expresó en el etiquetado XML con un atributo de las palabras con el nombre "sint".

USE LEXICAL ITEMS = NO
No se consideró la forma de las palabras, es decir, no se consideró el contenido de las palabras.

INTRODUCE VERBAL MARK = NO
No se introdujo marca de la posición del verbo en el patrón.

COLAPSE PATTERNS = YES
Se colapsarán los patrones, es decir, se identificarán adjuntos.

MAX ITERATION FOR FIND COLAPSE PATTERNS = FALSE
Se colapsarán los patrones mientras se cumplan las condiciones de colapsado.

MINIMAL ABSOLUTE VERBAL FREQUENCY = 100
No se consideraron los verbos que no tuvieran por lo menos 100 ocurrencias.

MINIMAL RELATIVE FREQUENCY TO CONSIDER PATTERN = 10
No se considerarán los patrones que co-ocurran menos de 10 veces con un verbo para formar parte del marco de subcategorización de ese verbo.

USE LIKELIHOOD RATIO TEST = YES
Este dará como resultado la certeza con que se acepta el patrón o "no_paso" en caso de no superar 90 % de confiabilidad. Y 'NO DECIDE' en caso de no darse las condiciones para realizar el test.

7.4. Descripción de los experimentos y Análisis de Resultados

Estudiamos los verbos **кушать** (comer, transitivo), **дать** (dar, transitivo con objeto indirecto) y **спать** (dormir, intransitivo), y nos centramos en los 10 patrones más frecuentes para cada uno de estos verbos. Estos tres verbos dan una buena muestra de cómo el sistema es capaz de identificar la estructura sintáctica de verbos con comportamientos muy distintos.

Veamos los patrones obtenidos por el sistema para el verbo **кушать** (comer):

Patrón	# ocurrencias	Test Likelihood %
['V', 'Nn']	5	99
['V', 'C']	5	95
['V', 'R']	4	no_paso
['V', 'Nn', 'C', 'Q']	3	95
['V', 'V', 'Nn', 'Nn']	3	99
['V', 'Nn', 'Na']	3	99,5
['Nn', 'C']	3	90
['V', 'Nn', 'Nn']	3	99
['V', 'R', 'Q']	2	95
['V', 'Nn', 'An']	2	99

Había pocas ocurrencias del verbo en el corpus, esto no era esperado ya que es un corpus de 500 MB (1.000.000 de palabras). El total de ocurrencias del verbo "кушать" en el corpus fue 137. Hubo una gran variabilidad entre los patrones, se encontraron 108 patrones distintos.

En la tabla anterior se puede observar que si el umbral del test de frecuencia está en 10 (el patrón ocurrió como mínimo 10 veces con el verbo en el corpus), ninguno de los 10 patrones más frecuentes para el verbo "кушать" lo pasaría.

Algo sorprendente fue que de los 10 patrones más frecuentes, 9 contenían verbos. Para entender este comportamiento no esperado se buscaron las oraciones que dieron origen a los patrones:
y efectivamente en las oraciones co-ocurría el verbo "кушать" con otros verbos, en construcciones como "querer comer", "coman coman", "callate y comé".

En contraste, no encontramos una cantidad importante de este comportamiento en el corpus del español porque se analizan por separado las diferentes oraciones, y las perífrasis verbales se analizan como un solo verbo.

Si analizamos los 10 patrones más frecuenes, vemos que los diferentes patrones que co-ocurren con el verbo **кушать** (comer) tienen diferente distribución en el corpus, por lo cual algunos, que ocurren mucho con el verbo pero relativamente poco en el resto del corpus, pasan el test de hipótesis de likelihood ratio, como por ejemplo ['V', 'Nn', 'Na']. En este patrón se observa el marco de subcategorización propio de los verbos transitivos: un sujeto (Nn, nombre nominativo) y un objeto directo (Na, nombre acusativo), y supera el test más estricto, con un umbral de 99.5 %. En cambio otros patrones, que son muy frecuentes en el corpus y por lo tanto poco específicos del verbo, no lo pasan, como por ejemplo ['V', 'R'], que es un patrón muy inespecífico, básicamente compuesto por un constituyente circunstancial (R, adverbio).

Si tratamos de recabar la misma información para el verbo "comer" en español del corpus Sensem, encontramos los 10 patrones de función sintáctica más frecuentes sin distinguir sentidos:

1. '(Suj. Elidido) [Obj Directo]': 18
2. '[Sujeto] + [Obj Directo]': 11
3. '(Suj. Elidido) [Circunstancial]': 10

4. '[Sujeto] + [Circunstancial]': 8
5. '(Suj. Elidido) [Obj Directo] + [Circunstancial]': 8
6. '(Suj. Elidido) [Circunstancial] + [Obj Directo]': 7
7. '(Suj. Elidido) [Circunstancial] + [Circunstancial]': 7
8. '[Sujeto]': 4
9. '[Sujeto] + [Obj Directo] + [Circunstancial]': 4
10. '[Circunstancial] + [Sujeto]': 3

En general no es fácil comparar porque el Sensem tiene mucha información (en el Sensem, se han estudiado las frases y etiquetado los circunstanciales que por lo general son un conjunto de palabras) y en el corpus ruso sólo hay información de las palabras...

Veamos qué son los circunstanciales:
Se denomina **complemento circunstancial** a la función sintáctica desempeñada por un sintagma adverbial, por un sintagma nominal, por un sintagma preposicional o por una oración subordinada que señale alguna circunstancia semántica de tiempo, lugar o modo al verbo de que es complemento, a veces incluso cantidad, causa, posibilidad, afirmación, negación o finalidad.
Sintagma adverbial es aquel en que el adverbio o gerundio desempeña función sintáctica de núcleo o palabra más importante y con más relaciones sintácticas.
Sintagma nominal, es el sintagma o grupo de palabras que forma un constituyente sintáctico maximal.
Sintagma preposicional o construcción preposicional esta construido por una preposición u otro tipo de adposición que funciona como núcleo sintáctico del sintagma preposicional que asigna caso al sintagma nominal o sintagma determinante que le sigue.
Oración subordinada o secundaria es una oración que depende de la preposición principal.

Quizás los constituyentes V, C, R se agruparían en circunstanciales, recordemos que V son verbos, C son conjunciones (incluyendo subordinantes) y R adverbios. Por lo tanto, estos son constituyentes susceptibles de ser parte de oraciones subordinadas y sintagmas adverbiales, es decir, buenos candidatos a adjuntos. Sin embargo, al no tener el corpus anotado con información sintáctica, este análisis es un poco especulativo.

Veamos los 10 patrones más frecuentes obtenidos por el sistema para el verbo **дать** (dar):

Patrón	# ocurrencias	Test Likelihood %
['Nn', 'Na', 'Nd']	66	99,5
['Q', 'Ng']	40	no_paso
['Nn', 'Nd']	39	99,5
['Nn', 'Na', 'Nd', 'C']	36	95
['Nn', 'Nd', 'V', 'Q']	32	99,5
['Nn', 'Q']	30	no_paso
['Nn', 'Na', 'Nd', 'V']	29	99,5
['Nd', 'Q']	28	no_paso
['Na', 'Nd']	26	95
['Nn', 'Nn', 'Nd']	24	95

En este caso, el verbo **дать** (dar) tenía muchas más ocurrencias en el corpus que el verbo "comer", por lo tanto los patrones también ocurrían con el verbo con mayor frecuencia y pasaban el test de frecuencias. Tampoco había tantas ocurrencias con patrones extraños, como los que contienen verbos.

Por ejemplo en el primer patrón **[Nn Na Nd]** está muy claro que es correcto, veamos porque:

Nn corresponde a un sustantivo nominativo, lo que en español podríamos decir que es el Sujeto, **Na** corresponde a un sustantivo acusativo, en español sería el Objeto Directo y **Nd** es el Objeto Indirecto.
Y vemos que en el Sensem que **[Sujeto] + [Obj Directo] + [Obj Indirecto]** pertenece al marco de subcategorización del verbo dar (vea los items 3, 7, 10, de la siguiente lista, ya no tuvimos en cuenta el orden para ejecutar el sistema).

También podemos ver que el patrón **[Na Nd]** se corresponde con el item 5 de la lista de los patrones que se encuentran en SenSem. (Sujeto elidido) significa que se ha omitido el sujeto en la frase, se induce por el contexto, como el Sensem posee información adquirida manualmente, agregaron esta información. Notar que el sistema no ve esta clase de elementos (ya que no estan en las oraciones).

Los 10 patrones de mayor significancia en la BD Sensem fueron:

1. '[Sujeto] + [Obj Directo]': 12
2. '(Suj. Elidido) [Obj Directo]': 9
3. '[Sujeto] + [Obj Indirecto] + [Obj Directo]': 8
4. '(Suj. Elidido) [Circunstancial] + [Obj Directo]': 6
5. '(Suj. Elidido) [Obj Indirecto] + [Obj Directo]': 6
6. '(Suj. Elidido) [Obj Directo] + [Obj Indirecto]': 5
7. '[Sujeto] + [Obj Directo] + [Obj Indirecto]': 5
8. '(Suj. Elidido) [Obj Directo] + [Circunstancial]': 4
9. '[Sujeto] + [Circunstancial]': 3
10. '[Obj Directo] + [Obj Indirecto] + [Sujeto]': 3

Veamos los 10 patrones más frecuentes obtenidos por el sistema para el verbo **спать** (dormir):

Patrón	Contador	Test Likelihood %
['V', 'C']	26	99,5
['V', 'Nn']	25	99,5
['Nn', 'R']	24	99,5
['V', 'R']	22	99,5
['V', 'Q']	18	90
['V', 'Nn', 'R', 'C']	15	99,5
['Nn', 'Q']	15	no_paso
['V', 'Nn', 'Nn', 'C']	14	99,5
['V', 'Nn', 'Nn', 'R']	12	99,5
['V', 'V']	12	90

Acá también vemos que 'V' aparece en 8 de los 10 patrones considerados.

En el Sensem vemos en el marco del verbo "dormir" están:

1. '(Suj. Elidido) [Circunstancial]': 28
2. '[Sujeto] + [Circunstancial]': 27
3. '[Sujeto]': 10
4. '(Suj. Elidido) [Circunstancial] + [Circunstancial]': 9
5. '[Circunstancial] + [Sujeto]': 6
6. '[Sujeto] + [Circunstancial] + [Circunstancial]': 4
7. '[Circunstancial] + [Sujeto] + [Circunstancial]': 4
8. '[Sujeto] + [Predicativo]': 2
9. '[Sujeto] + [Obj Directo]': 2
10. '(Suj. Elidido) [Predicativo]': 1

Por lo tanto, "dormir" es un verbo sin otros complementos obligatorios que el sujeto (es un verbo intransitivo), y suele ocurrir a menudo con complementos circunstanciales o que indican circunstancias de la acción de dormir, como los complementos predicativos.

En los patrones encontrados en el corpus del ruso podemos ver que en el 60 % de los patrones está Nn, lo que se corresponde con este comportamiento de verbo intransitivo. También vemos que hay 4 R, que son sonstituyentes de sintagmas adverbiales, y como dijimos anteriormente los sintagmas adverbiales son circunstanciales.

Las conclusiones en general de la ejecución en ruso, es que si bien no tenemos un gold standard para comparar, los resultados parecen corresponderse con lo que sabemos a priori sobre cada uno de los verbos estudiados. Es decir, los patrones identificados se corresponden con un comportamiento transitivo en el caso del verbo "comer", transitivo con objeto indirecto para "dar" e intransitivo para "dormir".

En el caso del verbo "comer", se dió que tenemos pocos datos y la naturaleza de los ejemplos del corpus son menos demostrativos de lo que esperaríamos, pero igualmente el verbo queda muy fuertemente asociado al patrón [Nn Na] (nominativo - acusativo, sujeto - objeto directo), característico de los verbos transitivos.

Capítulo 8

Conclusiones y Trabajo Futuro

8.0.1. Logros

En esta tesis hemos analizado el problema de adquisición automática a partir de corpus textuales de marcos de subcategorización asociados a piezas léxicas verbales. Hemos realizado una revisión del trabajo previo en el área, con especial atención a los trabajos para lenguas distintas del inglés. Hemos detectado los elementos relevantes para el abordaje del problema. Hemos diseñado, implementado y evaluado una herramienta altamente parametrizable e independiente de lengua que identifica los marcos de subcategorización asociados a los verbos a partir de la evidencia encontrada en un corpus dado, llamada IRASubcat.

Ventajas de la herramienta IRASubcat:

- Es gratis y de código abierto, con licencia GPL.
- Es independiente de lengua.
- Es multiplataforma.
- Es flexible - en cuanto a la entrada y la salida.
- Es parametrizable. Por lo cual se adapta a muchísimas opciones de procesamiento.
- No restringe los tipos ni cantidad de marcos, sino que se adapta a lo que se encuentra en el corpus.

Como resultado de esta tesis, además de la herramienta propiamente, se ha elaborado una publicación que se encuentra en proceso de revisión para ser publicada en un congreso internacional:

Altamirano, Ivana Romina, Alonso, Laura. 2010. IRASubcat, a highly parametrizable, language independent tool for the acquisition of verbal subcategorization information from corpus. *submitted*.

En el transcurso del trabajo, observamos las siguientes propiedades interesantes o criticables de algunas de las soluciones que elegimos implementar:

El test likelihood ratio

Es un test de hipótesis apropiado para varias tareas de PLN, pero se encontraron los siguientes problemas:

- El test no decide en los casos de que

  $$p1 < p2$$

  (siendo p1 la probabilidad del marco dado el verbo, y p2 la probabilidad del marco dado que no se dió el verbo).

Colapsador de patrones por identificación de adjuntos
El colapsador de patrones no encuentra los adjuntos tal que el mismo constituyente es adjunto y se encuentra 2 veces en el patrón. Tampoco identifica los adjuntos si éstos no aparecieron como los elementos menos frecuentes del patrón.

Como ventaja se puede decir que al ser conservador, no identifica adjuntos que no lo son.

8.0.2. Trabajo Futuro

Como futuras mejoras para la herramienta, nos planteamos lo siguiente:

- Una interfaz gráfica para el sistema.
- Un módulo que se encargue de marcar verbos en un corpus plano, según heurísticas ingresadas por el usuario.
- Agregar un módulo para agrupar clases de palabras - esto ayuda a reducir el data-sparseness.
- Agregar opción para ignorar valores de etiquetas.
- Implementar el test de hipótesis binomial, que se sumará al de likelihood ratio.
- Modificar el colapsador de patrones a fin de poder identificar adjuntos cuando el constituyente está en otra posición que no es la de hoja.También se podría permitir al usuario ingresar un umbral para la proporción de patrones que cumplan el test de adjuntos para colapsarlos.

Referencias

AG, B. I. . F. A. B. (Ed.) (2001). *Duden das Stilwörterbuch.* Dudenverlag.

Boguraev, B. (Ed.) (1987). *Alvey NL Tool Dictionary.*

Brent, M. R. (1993). From grammar to lexicon: Unsupervised learning of lexical syntax. pp. 243–262.

Chrupala, G. (2003). Acquiring verb subcategorization from spanish corpora. Master's thesis, Universitat de Barcelona.

de la Langue Française, I.Ñ. (Ed.). *Frantext online literary database.*

de Lima, E. (2002). The automatic acquisition of lexical information from portuguese text corpora. Master's thesis, Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart.

Dowty, D. (1979). *Word meaning and Montague grammar. The semantics of verbs and times in Generative Semantics and in Montague's PTQ: Synthese Language Library. Dordrecht: Reidel.*

Dunning, T. (1993). Accurate methods for the statistics of surprise and coincidence. *COMPUTATIONAL LINGUISTICS.*

Eckle-Kohler, J. (1999). Linguistic knowledge for automatic lexicon acquisition from german text corpora.

Fernández, A., G. V. e. I. C. (Ed.) (2004.). *Sensem: base de datos verbal del español.* (p. 155-163. ed.). G. de Ita, O. Fuentes, M. Osori.

Fillmore, C. J. (1968). "the case for case", in bach, emmon; harms, emmon, universals in linguistic theory, new york: Holt, rinehart and winston.

Georgala, E. (2003). A statistical grammar model for modern greek: The context-free grammar.

Gonzalez, A. O. (2004). Adquisició d'informació lèxica i morfosintàctica a partir de corpus sense anotar: aplicació al rus i al croat. Master's thesis, Universitat de Barcelona.

Grishman/Macleod/Meyers (Ed.) (1994). *COMLEX Sintax Dictionary* (1. ed.).

Hale, K. and S. Keyser (2001). Prolegomenon to a theory of argument structure. linguistic inquiry monograph 39. cambridge: Mit press.

Heid, S. K. (2003). A dutch chunker as a basis for the extraction of linguistic knowledge.

Hornby, A. S. and S. Wehmeier (Eds.) (1985). *Oxford Advanced Learner's Dictionary* (. ed.). Oxford University Press.

John., B. T. (1997). Automatic extraction of subcategorization from corpora. pp. 356–363.

Lancaster-Oslo/Bergen (1978). The lancaster-oslo/bergen corpus of british english, for use with digital computers. http://khnt.hit.uib.no/icame/manuals/lob/INDEX.HTM.

Manning, C. D. (1993). Automatic acquisition of a large subcategorization dictionary from corpora. pp. 235–242.

O'Donovan Ruth/Michael Burke/Aoife Cahill, J. v. G. W. (2005). Large-scale induction and evaluation of lexical resources from the penn-ii and penn-iii treebanks. Volume 31, pp. 329–365.

Prolo, K. A. A. (2002). Identifying verb arguments and their syntactic function in the penn treebank. pp. 1982–1987.

Rooth, C. G. (1998). Valence induction with a head-lexicalized pcfg.

Salmon-Alt, C. P. S. (2006). Automatic extraction of subcategorization frames for french.

Sampson, G. (Ed.) (1994). *The Susanne Corpus* (11 Aug. 1997. ed.). Oxford Text Archive.

Sandhaus, E. (Ed.). *New York Times*.

Santorini, M. P. M. B. and M. A. Marcinkiewicz (Eds.) (1993). *The Penn Treebank: a Large Annotated Corpus of English* (19:313-330. ed.). Computational Linguistics.

Schmid, H. (Ed.) (1994). *Probabilistic Part-of-Speech Tagging Using Decicion Trees*.

Schulte im Walde, S. (2000). Clustering verbs semantically according to their alternation behaviour. In *COLING'00*, pp. 747–753.

Sgall, J. H. H. P. P. (Ed.) (2001). *Prague Dependency Treebank* (1 ed.). Linguistic Data Consortium, Philadelphia.

Taylor, L. and Knowles (Eds.) (1988). *the SEC corpus: the machine-readable corpus of spoken English*. University of Lancaster, UK, Ms.

University, O. (Ed.) (2007). *The British National Corpus* (version 3 (BNC XML Edition). ed.). Distributed by Oxford University Computing Services on behalf of the BNC Consortium.

Waibel, U. A. D. A. E. T. G. (1993). The automatic acquisition of frequencies of verb subcategorization frames from tagged corpora. pp. 95–106.

Wauschkuhn, O. (1999). Automatische extraktion von verbvalenzen aus deutschen text korpora. Master's thesis, Universität Stuttgart.

WSJ (Ed.) (1994). *Wall Street Journal*.

Zeman, S. A. (2000). Automatic extraction of subcategorization frames for czech. pp. 691–697.

Glosario

A

adjuntos Adjunto es un sintagma no seleccionado por el predicado es siempre opcional., pág. 15.

algoritmo no supervisado Un algoritmo no supervisado descubre patrones y tendencias de los datos., pág. 16.

argumentos Argumento es un sintagma puede ser obligatorio por haber sido seleccionado por el predicado., pág. 15.

C

chunk Cada chunk contiene una cabecera que indica algunos parámetros como el tipo de chunk, comentarios, tamaño, etc. Inmediatamente hay un área variable de datos que son decodificados por el programa según los parámetros indicados en la cabecera., pág. 15.

complemento directo En sintaxis se llama complemento directo u objeto directo a la función que desempeña un sintagma nominal, un pronombre o una proposición subordinada sustantiva que es requerida de forma directa y obligatoria por un verbo transitivo., pág. 42.

D

data sparseness esparcimiento de datos, es cuando hay mucha diversidad de los datos y por lo tanto bajas frecuecias. En PLN son comunes., pág. 42.

E

expresiones regulares Una expresión regular, a menudo llamada también patrón, es una expresión que describe un conjunto de cadenas sin enumerar sus elementos., pág. 15.

F

f-score Es una medida que depende de la precisión y el cobertura cuya fórmula es:

$$F = 2 * (precision * cobertura)/(precision + cobertura)$$

., pág. 15.

H

HL-PCFG A Head-Lexicalised Probabilistic Context-Free Grammar. Es una gramática probabilística libre de contexto de encabezado lexicalizado., pág. 17.

I

internet Es un conjunto descentralizado de redes de comunicación interconectadas, que utilizan la familia de protocolos TCP/IP, garantizando que las redes físicas heterogéneas que la componen funcionen como una red lógica única, de alcance mundial., pág. 8.

L

lexicones El lexicón es el "diccionario" en el que se registran las palabras que conoce un hablante. Este "diccionario" especifica los rasgos idiosincráticos de las piezas léxicas (palabras). Algunos modelos gramaticales formales basan la generación de oraciones en el procesamiento de los rasgos de las unidades del lexicón. En estos modelos, el lexicón no es parte de la gramática, sino que proyecta sus rasgos a través de mecanismos inherentes a las gramáticas., pág. 8.

léxicos Las palabras utilizadas en una región específica, las palabras de un idioma., pág. 8.

M

morfológicos La morfología es la rama de la lingüística que estudia la estructura interna de las palabras para delimitar, definir y clasificar sus unidades, las clases de palabras a las que da lugar (morfología flexiva) y la formación de nuevas palabras (morfología léxica)., pág. 8.

S

semánticos El término semántica se refiere a los aspectos del significado o interpretación del significado de un determinado símbolo, palabra, lenguaje o representación formal. En principio cualquier medio de expresión (lenguaje formal o natural) admite una correspondencia entre expresiones de símbolos o palabras y situaciones o conjuntos de cosas que se encuentran en el mundo físico o abstracto que puede ser descrito por dicho medio de expresión., pág. 8.

sintácticos La sintaxis, una subdisciplina de la lingüística y parte importante del análisis gramatical, se encarga del estudio de las reglas que gobiernan la combinatoria de constituyentes y la formación de unidades superiores a éstos, como los sintagmas y oraciones. La sintaxis, por tanto, estudia las formas en que se combinan las palabras, así como las relaciones sintagmáticas y paradigmáticas existentes entre ellas., pág. 8.

sufijo Se denomina sufijo al morfema derivativo de las lenguas o afijo que se agrega después del lexema, raíz o tema de una palabra y antes de los morfemas constitutivos para añadirle a este una información suplementaria. La palabra nueva así formada se denomina palabra derivada, y al mecanismo por el cual se hace derivación., pág. 15.

T

test de hipótesis La teoría de test de hipótesis estadísticas es una metodología que sirve para determinar la importancia o la veracidad de ciertas hipótesis en problemas estadísticos., pág. 15.

transitivos La transitividad es una característica de ciertos verbos de poder tener dos participantes o argumentos nucleares: uno llamado usualmente sujeto agente y otro llamado objeto (paciente)., pág. 42.

trie Un trie es un caso especial de autómata finito determinista (S, Σ, T, s, A), que sirve para almacenar un conjunto de cadenas E en el que:

- Σ es el alfabeto sobre el que están definidas las cadenas;

- S, el conjunto de estados, cada uno de los cuales representa un prefijo de E;
- la función de transición: T : S x $\Sigma \rightarrow$ S; está definida como sigue: T(x, σ) = xσ si x, x$\sigma \in$ S, e indefinida en otro caso;
- el estado inicial s corresponde a la cadena vacía λ;
- el conjunto de estados de aceptación A $\subseteq$ S es igual a E. Su nombre procede del término inglés retrieval.
  Las ventajas principales de los tries sobre los árboles de búsqueda binaria (BST) son:
  - búsqueda de claves más rápida. La búsqueda de una clave de longitud m tendrá en el peor de los casos un coste de O(m). Un BST tiene un coste de O(logn), siendo n el número de elementos del árbol, ya que la búsqueda depende de la profundidad del árbol, logarítmica con el número de claves.
  - menos espacio requerido para almacenar gran cantidad de cadenas pequeñas, puesto que las claves no se almacenan explícitamente
  - mejor funcionamiento para el algoritmo de búsqueda del prefijo más largo.

, pág. 26.

V

valencias Los argumentos o valencias, son una serie de complementos del verbo para formar lo que se denomina sintagma verbal., pág. 16.