Extracción de instancias de una clase desde textos en lenguaje natural independientes del dominio de aplicación

Ph.D.(c) Juan Carlos Blandón Andrade

Universidad Nacional de Colombia

Facultad de Minas, Departamento Ciencias de la Computación y de la Decisión

Medellín, Colombia

2017

Extracción de instancias de una clase desde textos en lenguaje natural independientes del dominio de aplicación

Ph.D. (c) Juan Carlos Blandón Andrade

Tesis de investigación presentada como requisito parcial para optar al título de:

Doctor en Ingeniería – Sistemas e Informática

Director:

Ph.D. Carlos Mario Zapata Jaramillo – Universidad Nacional de Colombia

Comité Doctoral:

Ph.D. Sergio España Cubillo – Utrecht University – The Netherlands

Ph.D. Gerardo E. Sierra Martínez – Universidad Nacional Autónoma de México (UNAM)

Ph.D. Jairo H. Aponte Melo – Universidad Nacional de Colombia – Sede Bogotá

Universidad Nacional de Colombia

Facultad de Minas, Departamento Ciencias de la Computación y de la Decisión

Medellín, Colombia

2017

DEDICATORIA

*A **Dios** y a la **Virgen María** que son mi constante compañía espiritual para lograr todas mis metas.*

*Especialmente a mis Padres **Juan Blandón** y **Rufina Andrade** porque ellos han sido mi principal cimiento para lograr todas mis metas académicas y me han apoyado incondicionalmente a lo largo del Doctorado y de toda mi vida. Este triunfo es para ellos. ¡¡¡ Muchas Gracias Padres !!!!*

*A mis hermanos **Leidy** y **Harvey** que me han apoyado al inicio y durante el Doctorado, también me han brindado la oportunidad de crecer a su lado como persona y profesional.*

*A mi novia **Vivi** por brindarme todo su amor y por su compañía a lo largo de estos años sobre todo en los momentos difíciles de nuestra relación.*

Agradecimientos

Quiero expresar un agradecimiento muy especial al profesor y Director del Trabajo de grado, **Ph.D. Carlos Mario Zapata Jaramillo**, por brindarme la gran oportunidad de cursar mi proceso Doctoral bajo su orientación. Su enorme confianza, colaboración, paciencia, sus calidades académicas, profesionales y como persona, han contribuido en gran medida para que hoy pueda graduarme como Doctor. Sus consejos y forma de trabajo siempre estarán presentes en mi camino académico y profesional.

Docentes y compañeros de las distintas asignaturas correspondientes al plan de estudios del Doctorado en Ingeniería – Sistemas e Informática de la Universidad Nacional de Colombia Sede Medellín. Por todos sus valiosos aportes para el mejoramiento de mi desempeño académico y profesional.

Vicerrectoría de Investigación y Extensión de la Universidad Nacional de Colombia Sede Medellín, por intermedio del programa nacional de apoyo a estudiantes de posgrado para el fortalecimiento de la investigación, creación e innovación de la Universidad Nacional de Colombia 2013-2015, por su apoyo económico.

Universidad Católica de Pereira, un agradecimiento especial a la institución por su apoyo, a todos los funcionarios y compañeros que me han colaborado de una u otra manera para el desarrollo de mi Posgrado.

Resumen

Las ontologías en computación se incluyen en el mundo de la inteligencia artificial y constituyen representaciones formales de un área de conocimiento o dominio. Las ontologías permiten modelar el conocimiento mediante una estructura de conceptos relacionados, lo cual proporciona un vocabulario común y que es de vital importancia para compartir información. La ingeniería ontológica es la disciplina que se encarga del estudio y construcción de herramientas para agilizar el proceso de creación de ontologías desde el lenguaje natural y tiene tres etapas cruciales: aprendizaje de ontologías (*Ontology Learning*), población de ontologías (*Ontology Population*) y enriquecimiento de ontologías (*Ontology Enrichment*). La literatura especializada muestra gran interés por las tres etapas y, para desarrollarlas, utiliza distintos métodos como estadística, extracción de información, procesamiento de lenguaje natural, aprendizaje de máquina (*Machine Learning*) y combinaciones entre ellos. Sin embargo, algunos problemas subsisten, tales como la dependencia del dominio de aplicación, la carencia de métodos completamente automáticos y la carencia de identificación de instancias de atributos. En consecuencia, el problema que se aborda en esta Tesis Doctoral es la extracción automática de instancias desde el lenguaje natural, sin importar el dominio de aplicación, con el fin de contribuir con el proceso de población de ontologías. En esta Tesis Doctoral se propone un método computacional que utiliza técnicas de extracción de información y procesamiento de lenguaje natural para extraer instancias de una clase y generar como resultado un archivo con una ontología completa en formato OWL, utilizando la herramienta GATE (*General Architecture for Text Engineering*). Los resultados son prometedores, pues se logra crear ontologías desde cero automáticamente, sin importar el dominio de aplicación y con buenos niveles de *precision*, *recall* y *F-measure*.

Palabras clave: Web semántica, ontologías, población automática de ontologías, procesamiento de lenguaje natural, extracción de información, patrones GATE-JAPE.

Abstract

Ontologies in computation belong to artificial intelligence. Ontologies are formal representations of a knowledge area or domain. Ontologies can be used for modeling knowledge by using a structure of related concepts. Such structure provides a common vocabulary and it is crucial for sharing information. Ontological engineering is a discipline for studying and constructing tools for improving the process of ontology creation from natural language. Such a process has three crucial stages: ontology learning, ontology population, and ontology enrichment. The state of the art shows great concern with the three stages, which are developed by using methods like statistics, information extraction, natural language processing, machine learning, and combinations of them. However, some problems still remain—*e.g.,* dependence on the application domain, lack of automation, and lack of attribute instance identification. Consequently, in this Ph.D. Thesis we address the problem of automated extraction of instances from natural language—regardless of the application domain—in order to contribute to the process of ontology population. In this Ph.D. Thesis we propose a computational method by using information extraction and natural language processing technologies in order to extract instances of a class and to generate as an output a file with a complete ontology in OWL format. We use the GATE (General Architecture for Text Engineering) tool for implementing the method. The results are promising, since we automatically create domain-independent ontologies from scratch. Also, our method exhibits satisfactory levels of precision, recall and F-measure.

Keywords: Semantic Web, Ontologies, Automatic Ontology Population, Natural Language Processing, information extraction, Gate-Jape patterns.

Contenido

Pág.

Lista de figuras

Pág.

Lista de Tablas

Pág.

Abreviaturas

Abreviatura	Término
AOP	*Automatic Ontology Population* (Población Automática de Ontologías)
EI	Extracción de Información
GATE	*General Architecture for Text Engineering*
IA	Inteligencia Artificial
JAPE	*Java Annotation Patterns Engine* (Motor de Patrones de anotación de Java)
LHS	*Left-Hand-Side* (Regla lado izquierdo)
ML	*Machine Learning* (Aprendizaje de Máquina)
OE	*Ontology Enrichment* (Enriquecimiento de Ontologías)
OL	*Ontology Learning* (Aprendizaje de Ontologías)
OP	*Ontology Population* (Población de Ontologías)
OWL	*Ontology Web Language*
PLN	Procesamiento de Lenguaje Natural
RDF	*Resource Description Framework*
SGML	*Standard Generalized Markup Language*
SWRL	*Semantic Web Rule Language*
W3C	*World Wide Web Consortium*
XML	*eXtensible Markup Language*

Introducción

En la Inteligencia Artificial, los esfuerzos se enfocan en comprender cómo funciona la mente humana y, con base en ello, se intenta construir entidades inteligentes para sintetizar y automatizar tareas intelectuales [1]. Una de esas entidades son las ontologías, que pueden expresar una representación formal del conocimiento modelando un área común o dominio. Según Berners-Lee *et al.* [2], en computación una ontología puede expresar la regla: ***"Si un código de ciudad se asocia con un código de estado y una dirección usa ese código de ciudad, entonces esa dirección tiene asociado el código de estado"***. Como las ontologías permiten inferir conocimiento, se puede mejorar el funcionamiento de la web, porque las búsquedas se tornan más precisas y se pueden buscar páginas utilizando palabras clave que eviten la ambigüedad, haciendo mención a conceptos precisos. La información de una página web se puede ligar con las estructuras de conocimiento y con reglas de inferencia.

Las ontologías aparecen como una alternativa para organizar la información, utilizando un vocabulario común mediante una estructura de conceptos, los cuales representan propiedades comunes de las instancias de una categoría, disminuyendo la complejidad y contribuyendo en la solución de problemas. Las ontologías permiten modelar un conocimiento común, es decir, que tienen como objetivo captar conocimiento consensuado de forma genérica. Además, se pueden utilizar y compartir mediante aplicaciones de software y para grupos específicos de personas. Generalmente, las construyen de forma cooperativa diferentes grupos de personas en distintos lugares [3].

La ingeniería ontológica es una disciplina que contiene una serie de actividades relacionadas con el proceso de creación de ontologías, el ciclo de vida de la ontología, las metodologías, herramientas y lenguajes para construir ontologías [3]. En esa construcción existen tres etapas cruciales: el aprendizaje de ontologías (*Ontology Learning*), la población de ontologías (*Ontology Population*) y el enriquecimiento de

ontologías (*Ontology enrichment*). El aprendizaje de ontologías hace referencia a la adquisición (semi) automática de elementos como conceptos y relaciones relevantes entre ellos en un dominio. La población de ontologías se refiere a la inserción de instancias de conceptos y relaciones dentro de una ontología existente. El enriquecimiento de ontologías hace referencia a extender una ontología existente agregando nuevos conceptos, relaciones y reglas [4].

La literatura especializada muestra gran interés por las tres etapas mencionadas, pero esta Tesis Doctoral se enfoca en la población de ontologías. Para abordar la problemática se vienen utilizando varios enfoques. Los métodos estadísticos se basan en la información, es decir, en la distribución de las palabras en el corpus, para lo cual utilizan métodos estocásticos y probabilísticos. Estos permiten resolver ambigüedad de oraciones largas y procesar gramáticas que pueden generar muchos análisis posibles, entre los que se destacan el teorema de Bayes, la varianza, la distribución condicional, la distribución estándar y la probabilidad marginal, entre otros [5,6].

Los métodos de extracción de Información se basan en el análisis del lenguaje natural para luego extraer fragmentos de información de forma automática. En el proceso se toman textos como entrada, se procesan y se entregan como salidas los datos que se pueden mostrar directamente al usuario, guardarlos en una base de datos u hoja de cálculo para análisis posterior. Entre las técnicas que más se usan se encuentran el reconocimiento de entidades nombradas y la resolución de correferencia [7]. Los métodos de procesamiento de lenguaje natural se encargan de analizar y representar textos que ocurren naturalmente en uno o más niveles de análisis lingüístico. El propósito es lograr el análisis, representación o generación de texto, para lo cual se utiliza una serie de herramientas computacionales que buscan el procesamiento lingüístico a nivel morfológico, sintáctico y semántico [8,9].

Los métodos basados en aprendizaje de máquina se basan en técnicas para que el computador aprenda. Específicamente, se trata de un aprendizaje inductivo, que consiste en crear algoritmos que sean capaces de generalizar comportamientos y reconocer patrones con información suministrada en forma de ejemplos que se deben cargar al sistema, es decir, se trata de predecir el comportamiento futuro teniendo en cuenta el pasado [10].

Los métodos híbridos agregan nuevos algoritmos de razonamiento combinados con los métodos mencionados anteriormente. Las combinaciones más comunes son aprendizaje de máquina con procesamiento de lenguaje natural, o también extracción de información con procesamiento de lenguaje natural [11,12].

La revisión de literatura muestra que la mayoría de métodos de extracción de instancias de una clase que se desarrollaron hasta ahora arrojan buenos resultados en comparación con su promedio de *precision* y *recall*. Sin embargo, se requieren métodos donde el dominio sea independiente [13,14], que permitan que la extracción sea automática o semi-automática [15–17], que utilicen cualquier fuente de información [18,19] y que cuenten con buenos niveles en los criterios de *precision*, *recall* y *F-Measure* [12,20]. Todo ello con el fin de contribuir con el proceso de población automática de ontologías. Además, las instancias de atributos poco se identifican, afectando los valores de las métricas definidas.

En esta Tesis Doctoral se presenta un método computacional con una arquitectura *pipeline*, es decir, se requiere que termine un proceso para empezar otro. En este método se utilizan técnicas de extracción de información y procesamiento de lenguaje natural para extraer instancias de una clase y sus atributos desde textos escritos en lenguaje natural. El método incluye una serie de patrones sintácticos que se implementan en la herramienta GATE [21] y que permiten encontrar varias entidades ontológicas, construir la ontología y poblarla con instancias de clases y de atributos.

Los resultados obtenidos son prometedores, pues se logra construir un sistema automático que, teniendo como entrada un archivo con extensión txt o pdf, permite generar ontologías, poblarlas y luego generar como salida un archivo con extensión owl, el cual después se puede abrir en cualquier herramienta de desarrollo de ontologías. El sistema se valida utilizando doce dominios diferentes, obteniendo buenos niveles de *precision, recall* y *F-measure*.

La Tesis se organiza de la siguiente manera: en el Capítulo **1** se presenta el marco teórico de referencia, donde se aclaran los conceptos necesarios para la realización de la investigación; en el Capítulo **2** se realiza un recorrido por la literatura especializada de la

investigación, mostrando algunos trabajos relevantes en el área; en el Capítulo **3** se presentan el planteamiento del problema y los objetivos; en el Capítulo **4** se muestra la solución; en el Capítulo **5** se presenta la validación del trabajo; finalmente, en el Capítulo **6** se presentan las conclusiones de la investigación y el trabajo futuro.

1. Marco Teórico

1.1 Inteligencia artificial

Según Luger [22] la inteligencia artificial se puede definir como: ***"La rama de la ciencia de la computación que se ocupa de la automatización de la conducta inteligente"***. La inteligencia artificial se basa en principios teóricos que incluyen estructuras de datos usadas en representación del conocimiento, los algoritmos necesarios para aplicar ese conocimiento y los lenguajes de programación necesarios para su implementación. En otras palabras, la inteligencia artificial trata de emular la forma de razonamiento de la mente humana.

Esta disciplina es muy amplia y se enfoca principalmente en las siguientes áreas de estudio [23]:

- Búsqueda de soluciones.
- Sistemas expertos.
- Procesamiento de lenguaje natural.
- Reconocimiento de modelos.
- Robótica.
- Aprendizaje de máquina.
- Lógica.
- Incertidumbre y "lógica difusa".

1.2 Ontologías

1.2.1 Definición

Una definición de ontología dada por Gruber puede ser: ***"Una ontología es una especificación explícita de una conceptualización"*** [24]. Otra definición mejorando la

de Gruber y dada por Borst es: ***"Las ontologías son una especificación formal de una conceptualización compartida"*** [25].

En esta Tesis Doctoral se abordan las ontologías desde el campo computacional, no desde la filosofía. Se puede decir que una ontología es un tipo especial de objeto de información. En los sistemas de inteligencia artificial lo que existe es lo que se puede representar. Así, las ontologías son un medio para modelar formalmente la estructura de un sistema, es decir, las entidades y relaciones relevantes que surgen desde su observación y que son útiles para un propósito particular [26].

Una definición sobre ontologías teniendo en cuenta varios autores puede ser: ***"las ontologías tienen como objetivo capturar conocimiento consensuado de forma genérica y que se puede reutilizar y compartir mediante aplicaciones de software para diferentes grupos. Normalmente, las construyen cooperativamente diferentes personas en diferentes lugares"*** [3]. Esto quiere decir que la tarea principal que se asocia con las ontologías es definir un vocabulario común, el cual debe describir conceptos básicos y las relaciones entre ellos en un dominio específico, para que luego lo utilice cualquier sistema.

1.2.2 Componentes

Los componentes de las ontologías se dan de acuerdo con el dominio de interés y las necesidades de los desarrolladores de ellas. A continuación, se presentan algunos de los componentes principales de las ontologías [3].

- **Conceptualización:** Conjunto de conceptos, objetos, relaciones y restricciones que caracterizan un dominio.
- **Clases:** Son conjuntos de individuos. Son las encargadas de detallar los conceptos del dominio. La clase hace referencia a un conjunto de objetos y cada objeto en la clase es una instancia de la misma. Una clase cuyos componentes son clases, se denomina superclase o metaclase. Un ejemplo de clases en un dominio de viaje puede ser locaciones como clase general de la cual se desprenden las clases ciudades, pueblos; en ese mismo dominio puede existir la

clase transporte como clase general y aviones, automóviles, trenes como clases hijas.

- **Instancias o individuos:** Son objetos del dominio de interés; en otras palabras, son instancias de clases. Estos se pueden agrupar en clases. Un ejemplo de individuos o instancias en el dominio de viaje podrían estar un avión de Matricula AA7462 que pertenece a la clase Avión; Buenos Aires es un objeto que pertenece a la clase ciudad.

- **Propiedades (*Slots*):** Son relaciones binarias entre individuos y pueden ser Inversas, funcionales, transitivas y simétricas, entre otras. Un ejemplo de relaciones en el dominio viaje es la existente entre avión y transporte, donde se puede afirmar que avión es un medio de transporte y se pueden relacionar. Los *slots* también permiten almacenar diferentes clases de valores. Hay dos tipos de propiedades:

    1. Propiedades de objeto (*Object Properties*): Para ligar un individuo con otro individuo. Un ejemplo de propiedades de objeto podría ser que un piloto trabaja para una aerolínea, entonces se podría identificar una relación María trabaja para Avianca.

    2. Propiedades de tipo de datos (*Datatype Properties*): Para ligar un individuo con un valor. Un ejemplo para propiedades de tipo podría ser María tiene 25 años de edad. Allí se está ligando al objeto María con un valor que al mismo tiempo tiene un tipo que en este caso es entero. La relación podría verse María tiene edad 25.

- ***Frame*:** Es un objeto que incluye clases, instancias y relaciones.

- **Axiomas:** Permiten modelar condiciones que se cumplen siempre. Un axioma estructural establece condiciones que tienen que ver con la jerarquía de la ontología. Un axioma no estructural establece relaciones entre los atributos de un concepto y es específico de un dominio. En ejemplo de axioma en el dominio viaje podría ser una función que permita definir que una carretera conecta dos ciudades diferentes.

Las ontologías se utilizan en gran medida para mejorar la comunicación entre organizaciones, personas y entre las aplicaciones. Esto permite que los sistemas de cómputo puedan operar entre ellos para llegar a un razonamiento automático. Gracias a las ontologías y el conocimiento contenido en ellas, se pueden extraer datos de las páginas web de forma automática, para su procesamiento y generación de conclusiones [3].

1.2.3 Clasificación

Existen varias clasificaciones de ontologías. En la **Figura 1-1** se presenta una clasificación de acuerdo con el nivel de generalidad, es decir, el nivel de dependencia a una tarea particular o un punto de vista. En el *Top-level Ontology* se describen conceptos muy generales como acción, espacio, materia, entre otros y los cuales son independientes de un dominio particular. En los niveles de *Domain ontologies* y *task ontologies* se describe el vocabulario relacionado con un dominio genérico (automóviles) o tarea o actividad (diagnóstico o venta), con lo cual se logra especializar los términos introducidos en la ontología de alto nivel. Por otra parte, *Application ontology* permite describir conceptos dependiendo de un dominio o tarea particular que, frecuentemente, son especializaciones relacionadas con las dos ontologías [27].

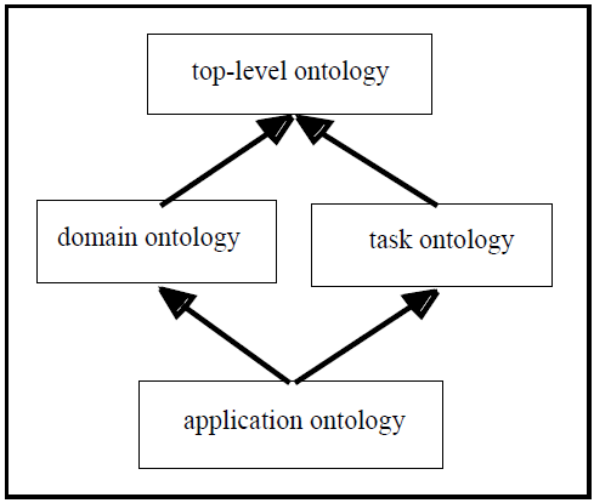


Figura 1-1: Clasificación de ontologías. **Fuente:** [27]

1.2.4 Lenguajes de ontologías basados en web

El auge de Internet posibilita la existencia de muchos lenguajes, con el fin de explotar las características de la web. Estos lenguajes, normalmente, se denominan lenguajes de

marcado y entre los más relevantes se encuentran XML (*Extensible Markup Language*), RDF (*Resource Description Framework*) y OWL (*Web Ontology Language*).

1.2.4.1 XML (*Extensible Markup Language*)

XML es un estándar de lenguaje de marcado y una versión reducida de SGML (*Standard Generalized Markup Language*), diseñado especialmente para documentos web. Este lenguaje permite a los diseñadores crear sus propias etiquetas personalizadas, lo cual permite la definición, transmisión e interpretación de datos entre aplicaciones y también entre organizaciones, lo que genera una mejor interoperabilidad [28].

El lenguaje tiene una serie de características que lo hacen relevante: es compatible con SGML, se utiliza en todo Internet y soporta una gran variedad de aplicaciones. Los documentos XML deben ser fáciles de crear y procesar, legibles para humanos y muy claros, con un lenguaje formal y conciso. Tiene algunos componentes como elementos, atributos, referencia a entidades, comentarios, instrucciones de procesamiento y *Prolog* que se utiliza para hacer un "*parsing"* de XML [3].

1.2.4.2 RDF (*Resource Description Framework*)

Es una recomendación del consorcio web W3C que se desarrolló para describir recursos web con metadatos. Este modelo de datos es equivalente a una red semántica, que es un gráfico que incluye un grupo de nodos y aristas con nombre. Los nodos representan conceptos, instancias de conceptos y valores de propiedades. Las aristas representan las propiedades de los conceptos o relaciones entre conceptos [3].

Un modelo de datos RDF tiene tres componentes: i) recursos, que se describen con expresiones RDF y se refieren como URIs (*Uniform Resource Identifiers*), opcionalmente con identificadores; ii) propiedades, también conocidas como predicados, que definen atributos o relaciones para describir un recurso; iii) declaraciones, que asignan un valor a una propiedad de un recurso específico y constan de sujetos, propiedades y objetos [3].

1.2.4.3 OWL (*Ontology Web Language*)

Es un formato recomendado por *W3C* y es el más expresivo para compartir ontologías en la web. Las declaraciones en *OWL* definen clases, propiedades, instancias, relaciones

inversas entre propiedades, clases disyuntas y un conjunto de axiomas, los cuales se utilizan para definir restricciones, que pueden ser tanto de cardinalidad de propiedades como combinaciones de clases [26].

El formato *OWL* se compone de tres niveles; i) OWL *Lite*, que tiene las características más comunes de *OWL*, se utiliza para crear taxonomías de clases y restricciones muy simples; ii) *OWL DL*, que es uno de los que más se utilizan a la hora de crear ontologías, pues incluye el vocabulario *OWL* de una forma completa y tiene capacidad de decibilidad, es decir, que el tipo de procesamiento es finito en la mayoría de las ocasiones; iii) *OWL Full*, es mucho más expresivo, pero no garantiza la computabilidad, eficiencia ni la decibilidad, es decir, las conclusiones pueden ser erradas o no realizadas [3].

1.2.5 Herramientas para el desarrollo de ontologías

La teoría sobre ontologías tiene sentido porque existen herramientas computacionales donde se puedan implementar esos modelos. Estas herramientas se pueden utilizar siempre y cuando haya la intervención de un ser humano que desarrolle los modelos, es decir, no se pretende comparar con el método que se desarrolla en esta Tesis, sino que se utilizarán (preferiblemente Protégé) para visualizar la ontología extraída desde el lenguaje natural. A continuación, se presentan las que más se usan en la actualidad.

1.2.5.1 Protégé

Creada en la Universidad de *Stanford*, es una herramienta libre para el desarrollo de ontologías y sistemas basados en conocimiento. Se desarrolló en Java y eso le da la ventaja de poder funcionar bajo diferentes sistemas operativos. Las aplicaciones que se construyen en *Protégé* se empelan para la resolución de problemas en dominios específicos. Esta herramienta emplea una interfaz gráfica de usuario muy integral porque permite la creación de una estructura de *frames* con clases, *slots* e instancias [29].

Al utilizar las vistas de la interfaz gráfica, los diseñadores de ontologías pueden crear clases, asignar propiedades a las clases y restringir las propiedades en ciertas clases. Usando ontologías resultantes, *Protégé* es capaz de generar automáticamente interfaces para crear individuos, debido a que se crea un formulario por cada clase utilizando las propiedades definidas [30].

1.2.5.2 Jena

Jena es un *Framework* que permite construir aplicaciones para la web semántica. Cuenta con una serie de librerías JAVA para que los desarrolladores puedan escribir código que se encargue de procesar *RDF*, *OWL*, *SPARQL* y sincronizado con las recomendaciones de la *W3C*. También incluye un motor de inferencia que se basa en reglas para razonar sobre ontologías *RDF* y *OWL* especialmente. Además cuenta con una serie de estrategias de almacenamiento para guardar tripletas *RDF* en el disco o en la memoria [30].

Una aplicación que se desarrolle para la web semántica debe realizar unas tareas comunes como: leer y analizar documentos RDF, crear y escribir documentos RDF, navegar y buscar en un grafo RDF, consultar en un conjunto de datos RDF utilizando SPARQL y realizar inferencias utilizando ontologías OWL. Estas funciones se pueden implementar utilizando el Framework Jena [31].

1.2.6 Usos de las ontologías

Las ontologías computacionales han sido objeto de estudio entre varios grupos de investigación y empresas, por eso se hace necesario conocer para qué son útiles. Los principales usos de las ontologías son [32]:

- Mejorar la búsqueda de información.
- Clarificar la estructura de conocimiento.
- Comprobar la validez de los datos.
- Organizar recursos multimedia.
- Programar agentes inteligentes.
- Permitir compartir conocimiento.
- Reducir la ambigüedad conceptual y terminológica.
- Realizar inferencia computacional.
- Reutilizar y organizar el conocimiento.

1.3 Ingeniería ontológica

La ingeniería ontológica es la disciplina que ***"se refiere al conjunto de actividades que conciernen al proceso de desarrollo de ontologías, el ciclo de vida de la ontología, y las metodologías, herramientas y lenguajes para la construcción de ontologías."*** [3]. En el proceso de construcción de ontologías existen tres etapas vitales: (i) el aprendizaje de ontologías, en el que se utilizan diferentes técnicas para conceptos y relaciones de las ontologías a partir de textos; (ii) la población de ontologías, que consiste en partir desde las ontologías para ir a los textos a buscar instancias de esos conceptos; (iii) el enriquecimiento de la ontología, que se puede definir como la ampliación de la ontología, es decir, se incorporan nuevos conceptos, relaciones y reglas y se realiza cuando el conocimiento del dominio no alcanza a explicar la información extraída del corpus [4].

1.3.1 Aprendizaje de ontologías

El aprendizaje de ontologías (*ontology learning*) es un proceso donde se aplican métodos y técnicas para construir ontologías desde cero y se utilizan diferentes fuentes de información que pueden ser estructuradas, semi-estructuradas o no estructuradas. Luego, se extraen términos, conceptos, relaciones y opcionalmente axiomas (reglas de inferencia) automáticamente. Para la realización de este proceso se utilizan diferentes técnicas como recuperación de información, minería de datos y procesamiento de lenguaje natural, entre otros. Al automatizar la construcción de ontologías, se pretende reducir tiempo y esfuerzo [33,34].

1.3.2 Población de ontologías

El proceso de población de ontologías (*ontology population*) consiste en insertar instancias de conceptos y relaciones dentro de una ontología existente. Actualmente, existen varias herramientas que permiten la extracción de instancias de conceptos e instancias de relaciones desde cualquier fuente de información y estas herramientas pueden ser semi-automáticas o automáticas [3].

En la **Figura 1-2** se presenta el proceso que se debe seguir para poblar una ontología. Al inicio se requiere una ontología que se puebla al final del proceso; también es necesario un corpus y un motor de extracción de instancias, el cual se encarga de localizar las

instancias de conceptos y relaciones en el corpus. Luego, se debe procesar el corpus utilizando el motor, con lo que se busca localizar conceptos y relaciones dentro del texto y se crea una lista con posibles instancias de conceptos y relaciones que después en un proceso definido se utilizan para poblar la ontología [4].

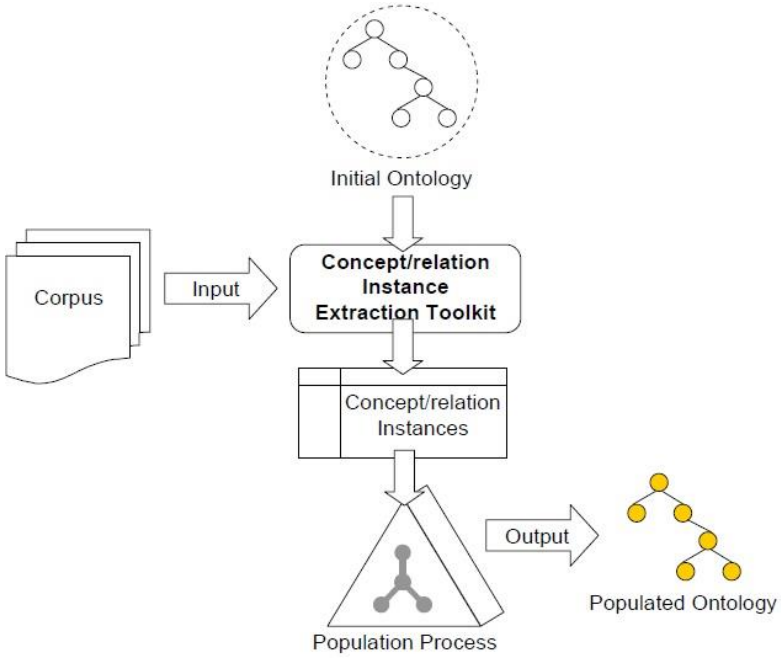


Figura 1-2: Proceso para Poblar una Ontología. **Fuente:** Adaptado de [4]

1.3.3 Enriquecimiento de ontologías

El proceso de enriquecimiento de ontologías (*ontology enrichment*) consiste en adicionar conceptos, relaciones y reglas. Esta etapa se realiza para ampliar el conocimiento, con el fin de explicar mejor en un futuro la información que se extrajo. Se debe tener en cuenta que, al encontrar nuevos conceptos y relaciones, la estructura de la ontología cambia y esto obliga a que se deba volver a realizar la población de ontologías. Este enriquecimiento se hace con el fin de minimizar la intervención del ser humano en el proceso de construcción de ontologías [4].

1.4 Técnicas para la población de ontologías

Existen distintos enfoques para abordar la población de ontologías, pero los más utilizados en la actualidad son los métodos estadísticos, de extracción de información, procesamiento de lenguaje natural, aprendizaje de máquina o combinaciones entre ellos.

Los métodos estadísticos para el procesamiento de lenguaje natural se basan en la información, es decir, en la distribución de las palabras en el corpus, para lo cual se utilizan métodos estocásticos, probabilísticos y estadísticos. Estos permiten resolver ambigüedad de oraciones largas y procesar gramáticas para generar análisis posibles. Entre los que más se usan se pueden destacar teorema de Bayes, varianza, distribución condicional, distribución estándar y probabilidad marginal, entre otros [5,6].

Estos métodos estadísticos permiten un acercamiento a los problemas que se presentan en el lenguaje y que están pendientes por resolver. Se debe tener en cuenta que la percepción humana es probabilística y que una de las formas de tratar de predecir eventos es la utilización de la estadística. Los métodos de extracción de información, procesamiento de lenguaje natural y aprendizaje de máquina encuentran en la estadística un soporte para su teoría y práctica [6].

Por otra parte, los métodos híbridos que realizan combinaciones entre los métodos existentes, de tal manera que se puedan optimizar los recursos de cómputo y aumentar la efectividad. La combinación se realiza entre los métodos estadísticos, extracción de información, procesamiento de lenguaje natural y aprendizaje de máquina. Las combinaciones más comunes son procesamiento de lenguaje natural con extracción de información, aprendizaje de máquinas con métodos basados en reglas, extracción de información con aprendizaje de máquinas. También existen otros métodos que realizan algoritmos para representar el conocimiento por medio de grafos [11], otros que integran reglas de adquisición de conocimiento lingüístico con recursos de inferencia [9] y finalmente sistemas de procesamiento de información algebraico combinado con un enfoque de multiagente basado en el análisis semántico del lenguaje natural [35].

1.4.1 Extracción de información

Los métodos de extracción de información son ***"una tecnología basada en el análisis del lenguaje natural para extraer fragmentos de información"*** [7]**.** Es decir, se refiere a extraer de forma automática fragmentos de información estructurada tal como entidades, relaciones entre entidades y la descripción de atributos de entidades desde fuentes de información no estructuradas [36]. El proceso toma textos como entrada, los procesa y entrega como salida datos que se pueden mostrar directamente al usuario o

guardar en una base de datos u hoja de cálculo, para análisis posterior. Con esta información se pueden realizar consultas de diferentes formas [7].

Esta labor abarca varios tipos de tareas, por ejemplo, la búsqueda de todos los nombres de personas que trabajan en una compañía o quizá la búsqueda de todos los asesinatos, incluyendo quién mató a quién, cuándo y dónde. Estas capacidades son cada vez más importantes porque permiten clasificar información desde enormes volúmenes de texto en línea y, finalmente, extraer la información específica que se requiere [8].

El proceso de extracción de información incluye algunas tareas; (i) el reconocimiento de entidades nombradas (*Named Entity Recognition*), que consiste en identificar nombres en un texto, los cuales se deben clasificar como personas, organizaciones y lugares, entre otros; (ii) la resolución de correferencia (*Coreference Resolution*), que identifica relaciones entre entidades y en la que se buscan frases que hacen referencia a objetos de clases semánticas específicas y se enlazan a frases que se refieren al mismo objeto; (iii) la extracción de relaciones, que consiste en identificar pares de entidades en una relación semántica específica; (iv) la extracción de eventos, en donde se identifican instancias de eventos de un tipo particular y los argumentos de cada evento [7,9].

Los sistemas de extracción de información se desarrollan para un dominio particular e incorporan las estructuras semánticas de ese dominio. La mayoría de trabajos en extracción de información se concentran en dominios que manejan grandes cantidades de textos, entidades repetidas y eventos del mismo tipo; esta información, a su vez, requiere su organización en una base de datos para su posterior utilización. Algunos dominios son los registros médicos y la literatura biomédica, nombres de genes y proteínas, generación de sinónimos, discurso desde vídeos, etc. [37].

1.4.2 Procesamiento de lenguaje natural

Algunas de las aplicaciones del procesamiento del lenguaje natural son la traducción automática, la recuperación de información, la extracción de resúmenes, los tutores inteligentes y el reconocimiento de voz, entre otros.

Los métodos de procesamiento de lenguaje natural hacen referencia a un conjunto de técnicas computacionales motivadas teóricamente para el análisis y representación de textos que ocurren naturalmente en uno o más niveles de análisis lingüístico, generalmente a nivel morfológico, sintáctico y semántico. El propósito es lograr el análisis, representación o generación de texto, para lo cual se utiliza una serie de herramientas computacionales que permiten procesar el lenguaje humano para realizar un rango de tareas o aplicaciones [8,9,38].

En el análisis morfológico se determina la categoría gramatical de una palabra que pertenece a una oración. Las unidades que constituyen las palabras se denominan morfemas, que son las mínimas unidades lingüísticas con significado. Según la estructura morfológica existen cuatro clases de lenguajes: aislados, aglutinadores, inflexionales y polisintéticos [39].

En el análisis sintáctico consiste en "Determinar las funciones de las palabras o grupos de palabras dentro de la oración" [39]. Este tipo de análisis también tiene funciones, la primera es determinar la estructura de las frases, para lo cual se deben revisar las relaciones entre las palabras, los modificadores de las palabras y se realiza un esquema en forma de árbol también llamado *Parsing*. La segunda consiste en regularizar la estructura sintáctica, donde se suprimen las palabras innecesarias y se determinan las equivalencias estructurales con diferencias en el tiempo. Existen dos tipos: análisis sintáctico por dependencias y análisis sintáctico por constituyentes.

El análisis semántico consiste en ***"asignar significados a las estructuras generadas por el analizador sintáctico, es decir se establecen correspondencias entre las estructuras sintácticas y cada palabra dentro de un dominio"*** [39]. Dentro del análisis semántico se han presentado varias propuestas para realizarlo. Uno de los primeros modelos fue el cálculo lambda, que se define como un modelo universal de computación usado ampliamente en semántica y ciencias de la computación para modelar comportamiento funcional de expresiones lingüísticas [8].

Otro modelo es HPSG (*Head-driven Phrase Structure Grammar*), el cual usa grafos dirigidos para representar toda la información lingüística como secuencias de características con tipos que toman valores de clases identificadas, donde esos valores

pueden, en sí mismos, ser estructuras de características [40]. Por su parte MRS (*Minimal Recursion Semantics*) hace referencia a que ***"En sí, no es una teoría semántica, pero puede ser más simple pensar en ella como un lenguaje de meta-nivel para la descripción de estructuras semánticas en algún lenguaje de objetos subyacente"*** [41].

MTT (*Meaning – Text Theory*) se encarga de ***"modelar la comprensión del lenguaje como un mecanismo que convierta los significados en los textos correspondientes y los textos en los significados correspondientes"*** [42]**.** Otra propuesta es la de grafos conceptuales, que son representaciones abstractas y lógicas con nodos llamados conceptos y relaciones conceptuales unidas mediante arcos [43]. Finalmente las interlinguas que ***"Son representaciones de textos en lenguaje neutral que se usan en traducción de máquina"*** [8]**.**

1.4.3 Aprendizaje de máquinas

Los métodos basados en aprendizaje de máquina (*Machine Learning*) hacen referencia a un aprendizaje inductivo, que consiste en crear algoritmos que sean capaces de generalizar comportamientos y reconocer patrones con información suministrada en forma de ejemplos que se deben cargar en el sistema, es decir, se trata de predecir el comportamiento futuro teniendo en cuenta datos del pasado [10,33].

Se pueden encontrar dos tipos de aprendizaje inductivo: supervisado y no supervisado. En el supervisado se predice la categoría adecuada para un ejemplo desde un conjunto de categorías representadas en un conjunto de etiquetas (datos históricos etiquetados). En el aprendizaje no supervisado se busca estructuras frecuentes y comunes dentro de los datos (datos históricos no etiquetados). Entre los enfoques más representativos se encuentran árboles de decisión, reglas de asociación, redes neuronales artificiales y redes bayesianas, entre otros [33].

1.4.4 Métodos basados en reglas

Los métodos basados en reglas utilizan conjuntos de reglas manualmente elaboradas que asocian características del texto a entidades. Luego se analizan los textos los cuales

posteriormente se anotarán y se regresan todas las entidades asociadas a cada parte del texto que se reconocen por medio de las reglas [44]. Estos sistemas deben exponer de una manera comprensible el conocimiento oculto en los datos, es decir, que la información tenga una estructura lógica para luego extraer conclusiones. Los conjuntos de reglas son útiles si son comprensibles, con alto nivel de precisión y no son muy numerosas [45].

Las reglas se usan para apoyar la toma de decisiones en clasificación, regresión y tareas de asociación. Existen varios tipos de reglas que pueden ser utilizadas para distintos tipos de conocimiento, la lógica proposicional clásica (*C-Rules*), reglas de asociación (*A-Rules*), lógica difusa (F-Rules), reglas de umbral (*T-Rules*), normas basadas en prototipos (*P-Rules*), entre otros. También se han desarrollado algoritmos para extraer reglas desde los mismos datos en los campos de estadística, aprendizaje automático, inteligencia computacional e inteligencia artificial [45].

1.5 Métodos de evaluación

Según Siau y Rossi [46] existen muchos métodos de modelado que los profesionales y los investigadores guardan celosamente. Se plantea la idea que el problema radica en la falta de técnicas estándar que permitan evaluar esos métodos. Es por esa razón que la evaluación de métodos de modelado se hace necesaria y para ello los autores realizan un compendio de varias técnicas usadas por profesionales e investigadores para evaluar métodos. Estas técnicas de evaluación se pueden clasificar en empíricas y no empíricas.

Entre los métodos no empíricos [46,47] se encuentran los siguientes:

Comparación de características: Uso de diferentes métodos para modelar el mismo dominio. De esta forma, se compara la manera como cada método aborda el mismo problema.

Metamodelo: Proceso de modelado que tiene un nivel de abstracción y lógica más alta que el proceso de modelado estándar. Captura información sobre los conceptos, formas de representación y usos de un método.

Evaluación Ontológica: Evaluación de las construcciones en los métodos existentes para hacerlos coincidir con construcciones ontológicas. Debe existir una correlación uno a uno entre las construcciones ontológicas y las construcciones de modelado.

Identificación de contingencia: Selección de un método de acuerdo con las contingencias del proyecto, tal como el problema bajo investigación y las personas que realizan la investigación.

Entre los métodos empíricos [46] se encuentran los siguientes:

Survey: Recolección de actitudes, opiniones y creencias utilizando cuestionarios.

Experimento de laboratorio: Manipulación de variables independientes y su posterior revisión de dependencias.

Experimento de campo: Desarrollo en un entorno natural. El investigador manipula las variables independientes y, al mismo tiempo, trata de controlar las variables más importantes que intervienen.

Caso de estudio: Uso de un sujeto particular, un grupo de sujetos u organización con uno o más métodos de modelado. El investigador observa sin necesidad de intervenir de alguna manera. Es decir, se trata de captar y comunicar la realidad de un ambiente particular en cualquier estado de tiempo.

1.5.1 Evaluación en extracción de instancias

En la fase de pruebas de un algoritmo de recuperación de información se suelen utilizar métodos para evaluarlo. Cuando se realiza un proceso de extracción de instancias de una clase desde un texto, o población de ontologías, se utilizan criterios como *precision*, *recall* y *F-measure* [48,49].

Precision: Mide la tasa entre el número de instancias correctamente extraídas (NICE) y el número de instancias extraídas (NIE). Véase la **ecuación (1.1)**.

$$P = \frac{NICE}{NIE} \quad (1.1)$$

Recall: Mide la tasa entre el número de instancias correctamente extraídas (NICE) y el número de instancias en el corpus (NIC). Véase la **ecuación (1.2)**.

$$R = \frac{NICE}{NIC} \quad (1.2)$$

F-Measure: Es un promedio ponderado de *precision* y *recall*. Cuando el valor se acerca a cero es el peor valor y uno es el mejor. Se puede decir que es la media armónica entre *precision* y *recall*. Véase la **ecuación (1.3)**.

$$F_1 = 2.\frac{Precision.Recall}{Precision + Recall} \quad (1.3)$$

1.6 Herramientas de Desarrollo

Cuando se piensa desarrollar un sistema que extraiga y procese información desde textos escritos en lenguaje natural, es necesario utilizar herramientas especializadas y que puedan brindar la oportunidad de procesar dicha información mediante la utilización de algún tipo de procesamiento lingüístico y que pueda generar algún tipo de salida. Debido a la naturaleza de esta Tesis, es de gran importancia la utilización de este tipo de herramientas, que tienen como las más representativas NLTK basado en Python y GATE basado en Java, entre otras existentes en la literatura.

1.6.1 NLTK (*Natural Language ToolKit*)

Es una herramienta implementada en Python. Tiene una gran cantidad de módulos para el procesamiento de lenguaje natural (véase la **Tabla 1-1**), por ejemplo acceso a corpora, tokenizadores, etiquetadores, analizadores, interpretación semántica, métricas de evaluación, probabilidad y estimación, entre otras [50].

Esta herramienta posee varias características: (i) simplicidad, debido a que no se requieren conocimientos avanzados de lenguaje natural; (ii) consistencia, pues cuenta con interfaces y estructuras de datos consistentes. (iii) extensibilidad, ya que se pueden

agregar fácilmente nuevos módulos e implementaciones alternativas. (iv) modularidad, para proveer componentes que se pueden usar de forma independiente [50]. La última versión de NLTK es la 3.2.1.

Tabla 1-1: Módulos de NLTK. **Fuente:** [50].

Tareas de Procesamiento de Lenguaje	***Módulos NLTK***	**Funcionalidad**
Acceso a corpora	*nltk.corpus*	*Standardized interfaces to corpora and lexicons*
Procesamiento de Cadenas	*nltk.tokenize, nltk.stem*	*Tokenizers, sentence tokenizers, stemmers*
Palabras Juntas *ej: disk drive (Collocation discovery)*	*nltk.collocations*	*t-test, chi-squared, point-wise mutual information*
Etiquetado gramatical	*nltk.tag*	*n-gram, backoff, Brill, HMM, TnT*
Clasificación	*nltk.classify, nltk.cluster*	*Decision tree, maximum entropy, naive Bayes, EM, k-means*
Extracción de entidades *(Chunking)*	*nltk.chunk*	*Regular expression, n-gram, named entity*
Analizadores *(Parsing)*	*nltk.parse*	*Chart, feature-based, unification, probabilistic, dependency*
Interpretación semántica	*nltk.sem, nltk.inference*	*Lambda calculus, first-order logic, model checking*
Métricas de Evaluación	*nltk.metrics*	*Precision, recall, agreement coefficients*
Probabilidad y estimación	nltk.probability	*Frequency distributions, smoothed probability distributions*
Aplicaciones	nltk.app, nltk.chat	*Graphical concordancer, parsers, WordNet browser, chatbots*

1.6.2 GATE (*General Architecture for Text Engineering*)

Creada en lenguaje Java, es una infraestructura para desarrollar y desplegar componentes de software que procesan lenguaje humano. Permite procesar cualquier tipo de textos sin importar su forma y tamaño. Es un software libre y se utiliza para procesar el lenguaje natural. También, permite trabajar varios idiomas y es muy versátil [51].

Las funciones principales de GATE son [51]:

- Modelado y persistencia de estructuras de datos especializadas.
- Medición y evaluación comparativa.
- Visualización y edición de anotaciones, ontologías y otros.
- Uso de un lenguaje basado en máquinas de estados finitos, que permite la creación de reglas como expresiones regulares (JAPE).

- Extracción de instancias de entrenamiento para aprendizaje automático.

Así mismo, GATE cuenta con tres tipos de recursos [51]:

Language Resource **(LR):** Representa lexicones, corpora, ontologías, tesauros o diccionarios.

Processing Resource **(PR):** Se pueden desarrollar algoritmos para analizadores sintácticos (*parsers*), generadores, lematizadores, traductores y reconocedores de discurso en un archivo JAR.

Visual Resources **(VRs):** Este recurso se puede utilizar para construir interfaces gráficas de usuario.

GATE permite la interacción con otras herramientas como WordNet y el analizador de Stanford, Protégé, entre otros. También se puede utilizar como una API en Java y se puede llamar desde allí, lo que permite crear aplicaciones de escritorio y web. Finalmente, la arquitectura sirve para la integración de módulos orientados a la resolución de problemas de procesamiento de lenguaje natural. La última versión de GATE es la 8.1.

1.6.2.1 JAPE (*Java Annotation Patterns Engine*)

GATE cuenta con un lenguaje que permite reconocer entidades en un texto determinado utilizando expresiones regulares que se denomina JAPE. Las reglas se deben expresar en archivos con extensión "*JAPE*" y deben tener una sintaxis que debe respetar. Luego, GATE permite crear los autómatas de estado finito para el total de archivos definidos y, de esta forma, reconocer expresiones. JAPE tiene una gramática la cual consiste en una serie de reglas patrón/acción. La parte izquierda (LHS, *Left-hand-side*), describe el patrón que se define para la expresión y es posible utilizar operadores de expresiones regulares [51]. La acción se expresa con la parte derecha de la regla (RHS, *right-hand-side*) y, básicamente, se utiliza para manipular anotaciones. Entonces, si se cumple un patrón del lado izquierdo de la regla, se debe cumplir la acción que se define para ese patrón. La parte derecha de una regla puede contener código Java. Para lograr que se ejecuten estos patrones, es necesario contar con un transductor (define alfabeto de entrada y

salida), que en la actualidad se denomina “JAPE-Plus *Transducer*” y, aunque en el pasado existieron otras versiones, los creadores mencionan que esta versión es mucho más eficiente y rápida [51].

En la **Figura 1-3** se presenta una regla en lenguaje JAPE. La regla se utiliza para encontrar nombres de aeropuertos en un texto en lenguaje natural. En todas las reglas escritas en JAPE, es necesario definir un encabezado (letra color negro). Primero se da un nombre a la fase, para el caso del ejemplo se llama “*Airport*”; luego se debe definir qué tipos de anotaciones se requieren como entrada, en el caso del ejemplo las anotaciones tipo “*Token*” y “*Lookup*”. Después, se debe definir el método que se debe aplicar en el caso de que las reglas se superpongan. Existen cinco opciones: *“appelt”*, *“first”*, *“once”*, *“brill”*, *”all*” [51]. Posteriormente, se debe definir la parte izquierda de la regla (texto color rojo), en la cual se define el nombre de la regla, una prioridad para evitar ambigüedades con otras reglas y el patrón a utilizar (para el ejemplo planteado, se busca el nombre de una ciudad en la lista “*city*”, la otra parte del patrón busca la palabra “*Airport”).* Entonces, si se encuentra un nombre de ciudad más la palabra “*Airport*”, se asigna la etiqueta “instAirport”. Teniendo en cuenta que se cumplió el patrón, se activa la parte derecha de la regla (texto color azul), para el ejemplo se crea una etiqueta llamada “*Airports*”, la cual recibe el texto que se almacenó en “*instAirport*”.

```
Phase:Airport
Input:  Token Lookup
Options: control = appelt

Rule: AirportRule
Priority: 35
(
{Lookup.minorType == city}
{Token.string == "Airport"}
):instAirport
-->
:instAirport.Airports = {rule = "AirportRule"}
```

Figura 1-3: Ejemplo de código en JAPE. **Fuente:** Elaboración propia

1.6.3 OWLIM (OWLMemShemaRepository SAIL)

Es un moderno repositorio semántico y de alto rendimiento, actúa como una base de datos, pero realmente añade capacidades de ontología, con lo cual permite realizar consultas a través del lenguaje SPARQL. Una ontología *OWLIM* puede almacenar gran cantidad de entidades ontológicas y se pueden realizar inferencias lógicas mediante el uso de reglas. Cada regla tiene un conjunto de premisas, que en conjunto definen el cuerpo de la regla. Las premisas son declaraciones RDF, que pueden contener variables libres. *OWLIM* permite el acceso a la información mediante el uso del *framework* JENA [52].

2. Revisión de Literatura

El interés por la extracción de información desde textos en lenguaje natural se viene trabajando desde hace algunos años. El hecho de poder extraer cualquier tipo de información desde el lenguaje natural es muy atractivo para tratar de automatizar procesos y evitar tareas largas y complejas. La extracción de instancias de clases (para este trabajo población de ontologías) se utiliza para lograr distintos objetivos y se inició aproximadamente desde los años ochenta y en la actualidad se sigue trabajando en el tema.

Abbott [53] presenta una técnica para desarrollar programas desde descripciones informales pero precisas en idioma inglés. La técnica demuestra cómo derivar tipos de datos (categoría de seres o cosas) desde sustantivos comunes, variables desde verbos y atributos y estructuras de control desde sus equivalentes en inglés. La principal contribución de este trabajo es la relación propuesta entre sustantivos comunes y tipos de datos. La idea es capturar estos elementos y transformarlos en un programa escrito en ADA. El artículo presenta una discusión de cómo hacer la transformación entre sintagmas nominales, tipos de datos y objetos.

Contreras [15] propone una arquitectura de adquisición de contenido para la web semántica, que provee un marco conceptual para desarrollar sistemas de procesamiento de contenido web y mapear contenidos semánticamente anotados, lo que permite el procesamiento por medio de agentes de software y aplicaciones de web semántica. También genera una ontología automáticamente, extrae instancias desde textos, asigna instancias a clases y extrae valores de atributos. La arquitectura acepta como entrada archivos TXT para luego utilizar técnicas de procesamiento de lenguaje natural.

Pasca [18] propone un método para adquirir entidades nombradas en categorías arbitrarias utilizando patrones léxico-sintácticos. También, hace referencia al refinamiento

de una consulta en una búsqueda web. Las categorías de nombres recogidas se fusionan eficazmente y luego resumen las relaciones semánticas detectadas en los documentos iniciales. Se extraen en pares. Por ejemplo: NombreNavegador, Google. Luego, se utilizan los patrones léxico-sintácticos para extraer las instancias. Esos patrones se obtienen de documentos de entrenamiento automáticamente, que constituyen las reglas que se deben cumplir antes de hacer la extracción de las instancias.

Geleijns y Korst [16] presentan un método que utiliza patrones hechos a mano y los construyen a la medida para las clases y relaciones consideradas. Los patrones se consultan en Google, donde los resultados se utilizan para buscar otras instancias. Las instancias que se encuentran se utilizan dentro de los patrones, de tal forma que el algoritmo puede poblar la ontología, al utilizar unas pocas instancias de una ontología parcial dada. También se debe construir una ontología parcial en forma de tupla. Luego, se alimenta el sistema con pocas instancias de clases y relaciones escritas a mano y el sistema busca en la web qué instancias diferentes puede encontrar. Así, se logran poblar las clases y las relaciones.

De Boer *et al.* [54] presentan una propuesta para la extracción de instancias de relaciones, por ejemplo, la relación artista-estilo artista. También, se trabaja un dominio de fútbol. Teniendo como base la ontología, se pueden extraer las instancias de las relaciones desde un corpus, que en este caso es la web. Específicamente, el método necesita dos conjuntos de instancias de clases Ci y Cj. Después, toma una instancia i de Ci, con la cual se eligen los documentos desde la Web. Posteriormente, se utilizan todas las instancias de Cj para saber cuántas existen, en cada uno de los documentos encontrados. De esta forma, se obtienen las instancias de las relaciones entre Ci y Cj.

Yoon *et al.* [55] proponen un método automático para población de ontologías con datos en formato estructurado. Las instancias se extraen desde páginas web utilizando *wrappers* o sentencias mediante técnicas de procesamiento de lenguaje natural. El método requiere una ontología e instancias semillas y se extraen desde documentos semi-estructurados o no estructurados. El método tiene una precisión del 98%.

Talukdar *et al.* [11] presentan un algoritmo de propagación de etiquetas semi-supervisado y el cual utiliza un gráfico llamado "*Adsorption*", después utilizan fuentes estructuradas y no estructuradas de información para adquirir clases etiquetadas y las instancias en un dominio abierto. Así, construyen un gráfico donde cada nodo representa tanto una instancia o una clase y existe un puente entre un nodo instancia y un nodo clase, siempre y cuando la instancia pertenezca a esa clase. Esta herramienta requiere una clase y cinco instancias semillas que se utilizan para evaluar el texto y extraer instancias, las cuales permiten construir el gráfico que finalmente encuentra otras instancias que ayudan a etiquetar las clases a las que pertenecen.

Manine *et al.* [56] presentan una arquitectura para integrar ontologías en el dominio biomédico. La entrada del sistema debe partir de documentos muy especializados para poder entrenarlo y después se logra extraer instancias de la web de forma automática. Se utilizan técnicas de extracción de información y aprendizaje de máquina. Finalmente, se obtienen buenos niveles de *precision* y *recall*.

Ruiz-Martínez *et al.* [57] presentan un *framework* que procesa textos mediante herramientas de procesamiento de lenguaje natural. Realizan las pruebas con la ontología llamada "*Travel.owl",* la cual se descarga desde la página de Protégé y a la cual realizan algunos cambios. Utilizan una página web para extraer instancias pertenecientes a la clase Hotel que se encuentra dentro de la ontología que se trabaja.

Danger y Berlanga [58] utilizan una ontología de referencia, reconocedores de entidades y desambiguadores de entidades para crear y combinar adecuadamente un conjunto inicial de instancias. El análisis exhaustivo y experimentación de la propuesta se llevan a cabo en una variedad de escenarios de aplicación. En el proceso, se define una ontología en *OWL* que tiene conceptos y relaciones. Existen lexicones que describen las reglas léxicas, para después identificar conceptos y relaciones en el texto. Luego de extraer entidades, se define un conjunto de instancias inicial que, mediante el uso de reglas de inferencia, genera finalmente un conjunto de instancias complejas que definen semánticamente el documento de acuerdo con la ontología dada.

Faria y Girardi [49] presentan una propuesta para semi-automatizar la población de ontologías desde textos. Utilizan técnicas de Procesamiento de Lenguaje Natural (PLN) y

Extracción de información (EI), para clasificar instancias de ontologías. El proceso tiene dos fases:

(i) La primera fase realiza la extracción y clasificación de instancias que, a su vez, realiza tres tareas. La primera tarea es el análisis de corpus, mediante el cual se estructura el corpus y se realizan tres actividades (análisis morfoléxico, que identifica las categorías gramaticales, reconocimiento de nombre de entidades, que identifica nombre de personas, organizaciones o lugares y la identificación de correferencia que identifica las correferencias de pronombres y correferencias nominales). La segunda tarea es la especificación de reglas de clasificación y extracción, donde el usuario se basa en la ontología y patrones léxico-sintácticos definidos previamente, para generar un conjunto de reglas de extracción. La tercera tarea es la extracción y clasificación de instancias donde se utilizan las reglas de la tarea previa.

(ii) La segunda fase es la representación de instancias, en la cual se realizan dos tareas (el refinamiento de instancias y la población de la ontología). Los autores mencionan que están evaluando las ventajas de combinar técnicas PLN con *Soft computing*.

Schlaf y Remus [59] presentan un *framework* (véase la **Figura 2-1**) para aprender categorías y sus instancias mediante características contextuales. Su *framework* se basa en el uso de textos en lenguaje natural como ejemplos de entrenamiento. Consta de tres pasos: (i) aprendizaje de reglas desde los textos de ejemplo; (ii) selección de las reglas de alta calidad, por medio de dos filtros (el primer filtro tiene en cuenta el número de ocurrencias de la regla y el segundo filtro toma dos características no independientes); (iii) identificación de nuevas instancias de la categoría teniendo en cuenta las reglas de filtrado, que se basan en cuatro categorías (nombre, apellido, profesión y ciudad), que permiten ubicar palabras (profesor, ingeniero o abogado) para detectar automáticamente que son instancias de una clase (en este caso, profesión).

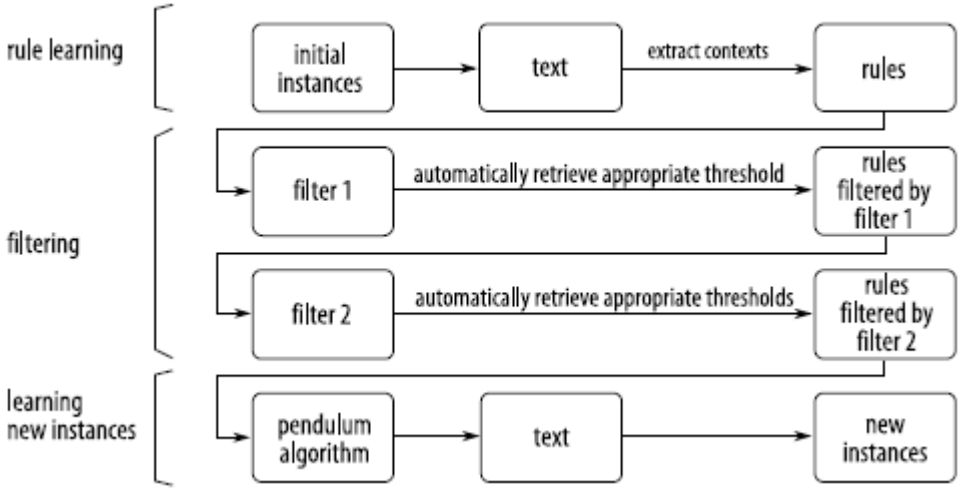


Figura 2-1: Descripción de *framework* de 3 pasos. **Fuente:** [59]

IJntema *et al.* [19] proponen un método que utiliza reglas para aprender instancias de ontologías desde textos, con el fin de contribuir con el proceso de población de ontologías. Las reglas léxico-semánticas explotan las capacidades de inferencia de las ontologías. Este sistema necesita ontologías del dominio a trabajar, para luego definir patrones léxico-semánticos. Con estas herramientas se procesa el documento para evaluar qué instancias se pueden extraer de páginas web de noticias.

Ruiz-Martinez *et al.* [60] presentan una metodología para la población de ontologías del dominio biomédico. El sistema se alimenta con una ontología de dominio biológico, enriquecida con instancias de textos en lenguaje natural. El proceso tiene tres capas (véase la **Figura 2-2**): (i) las ontologías de nivel superior, que definen las relaciones semánticas básicas a mapear, dentro de recursos que permiten etiquetar roles semánticos; (ii) la ontología del dominio a poblar, que se relaciona con el modelo ontológico; (iii) la ontología del dominio poblada, que se puebla mediante los modelos ontológicos y recursos lingüísticos. Los autores utilizan procesamiento de lenguaje natural y logran extraer instancias en el dominio biomédico y asignan instancias a clases automáticamente. Finalmente, obtienen buenos resultados de *recall* y *precision* en ese dominio.

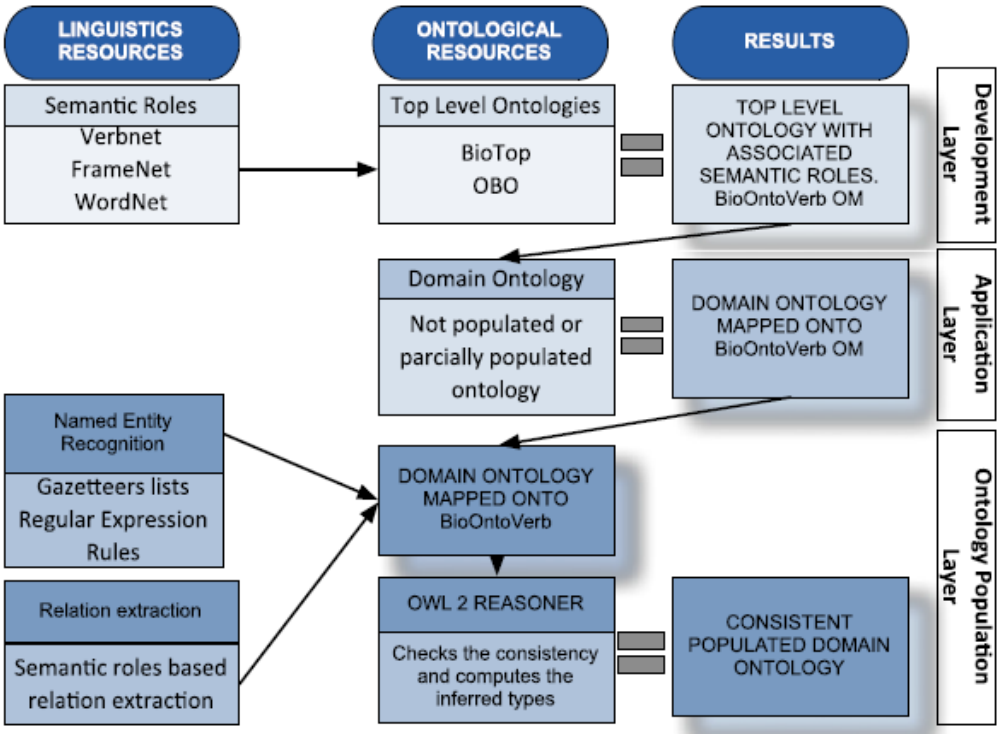


Figura 2-2: Arquitectura del *framework* BioOntoVerb. **Fuente:** [60]

Faria *et al.* [17] proponen un proceso para la población automática de ontologías desde textos. El proceso aplica procesamiento de lenguaje natural y técnicas de extracción de información para adquirir y clasificar instancias de ontologías. Es un paso inicial hacia la utilización de una ontología para generar reglas automáticamente desde ella, extraer instancias desde textos y clasificarlas en las clases de la ontología. Las reglas se generan a partir de ontologías de cualquier dominio para lograr el objetivo de independencia del dominio. El proceso tiene tres fases: (i) identificación de instancias candidatas; (ii) construcción de un clasificador; (iii) clasificación de instancias. El sistema se prueba en los dominios legal y turismo.

De Araujo *et al.* [12] proponen una metodología que permite poblar ontologías con instancias de eventos. La principal contribución se relaciona con la exploración de la flexibilidad de reglas lingüísticas y la representación del dominio de conocimiento mediante su manipulación e integración con un sistema de razonamiento. Los documentos a procesar se deben tratar con un programa de análisis lingüístico profundo (PALAVRAS) y luego representar en *OWL* con el modelo de datos POWLA. Para que luego se puedan usar reglas lingüísticas y los conceptos de la ontología del dominio. La

gran cantidad de información *OWL* que se genera sirve para hacer inferencias lógicas. Como salida se obtiene la ontología con las reglas lógicas y la ontología del dominio. Con todos esos elementos se utiliza un razonador, que permite extraer las instancias.

Sadoun *et al.* [61] presentan un enfoque que se centra en la identificación de instancias de propiedades mencionadas en textos, se utilizan reglas de extracción que se obtienen desde rutas sintácticas recurrentes y se vinculan términos que denotan conceptos e instancias de propiedades. El proceso requiere una ontología del dominio al igual que un corpus de entrenamiento. Con esos elementos se definen reglas de extracción, con el fin de extraer instancias de propiedades mencionadas en los textos. Las reglas explotan conocimiento léxico, sintáctico y semántico. Finalmente, los autores demuestran que con esa información pueden extraer instancias de clases implícita o explícitamente.

Ríos Alvarado [62] presenta la generación de ontologías, que incluyen axiomas de clases e instancias de manera automática a partir de textos en idioma inglés. Se utilizan técnicas como procesamiento de lenguaje natural, algoritmos de agrupamiento y extracción de información. Obtienen ontologías que incluyen conceptos, relaciones jerárquicas, axiomas e individuos. Finalmente, construyen buenas ontologías, las cuales comparan manualmente con la ontología de referencia llamada ***goldstandard***. En cuanto a las instancias obtienen resultados de *precision* del 56.8%.

Faria *et al.* [13] presentan un método (véase la **Figura 2-3**) de dominio independiente y se ajustan algunas partes del enfoque para mejorar los valores de *precision* y *recall*. Como entrada es necesario un corpus y una ontología vacía para realizar la población. El método cuenta básicamente con tres tareas: (i) identificación de instancias candidatas, mediante técnicas de procesamiento de lenguaje natural (*NLP*) y modelos estadísticos (*SMIR*); (ii) construcción de un clasificador, por medio de una herramienta de extracción de información (*IE*) y otra de aprendizaje de máquina (*ML*), en donde la función de la tarea consiste en seleccionar clases, propiedades y relaciones, además de seleccionar los disparadores y generar reglas; (iii) clasificación de instancias, que necesita como entradas el clasificador y el corpus anotado, para luego utilizar procesamiento de lenguaje natural (*PLN*) y aprendizaje de máquina (*ML*) para asignar las instancias a las clases, propiedades y relaciones y, finalmente, obtener como resultado la ontología poblada. Por otra parte, para lograr la independencia del dominio, generan un clasificador

desde la ontología procesada, de modo que, sin importar la ontología de entrada, se puebla desde documentos en lenguaje natural. Las pruebas se realizan bajo los dominios legal y turismo con buenos resultados.

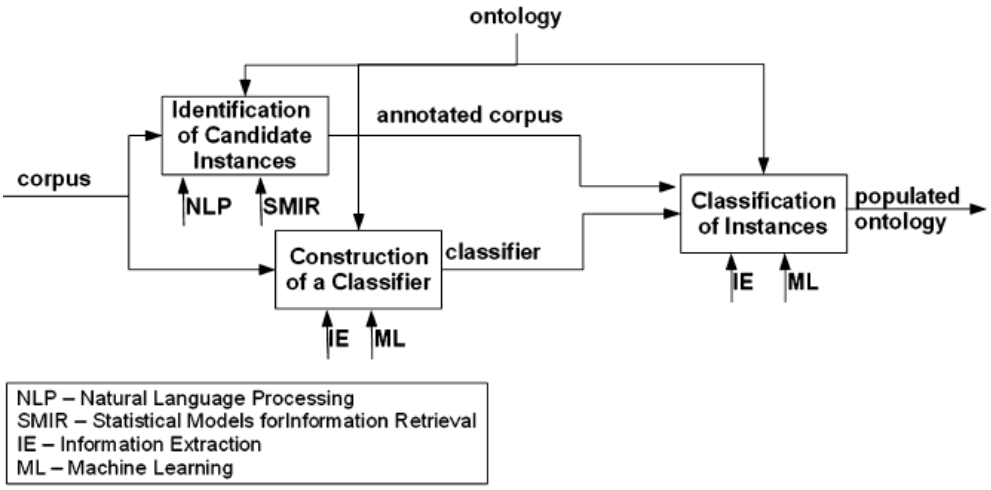


Figura 2-3: Un proceso genérico para población de ontologías. **Fuente:** [13]

Lima *et al.* [14] presentan un sistema (véase la **Figura 2-4**) que se basa en programación lógica inductiva (PLI) y, automáticamente, induce reglas de extracción simbólicas, que se utilizan para poblar un dominio de ontología con instancias de clases. El método explota la similitud semántica y tiene cuatro fases: (i) la recuperación del corpus, donde se recuperan oraciones desde la web para construir un corpus de trabajo (patrones *Hearst*), además de que el usuario elige una clase desde una ontología de dominio y después el sistema recupera algunos documentos con la elección del usuario; (ii) el pre-procesamiento del texto, donde se realiza un análisis léxico-sintáctico por medio del analizador de Stanford y se mide semánticamente la distancia entre la clase y las instancias candidatas mediante Wordnet; (iii) el mejoramiento de las reglas que se deben aplicar y las cuales se encuentran en la base de conocimiento; (iv) la aplicación de las reglas y la extracción de las instancias.

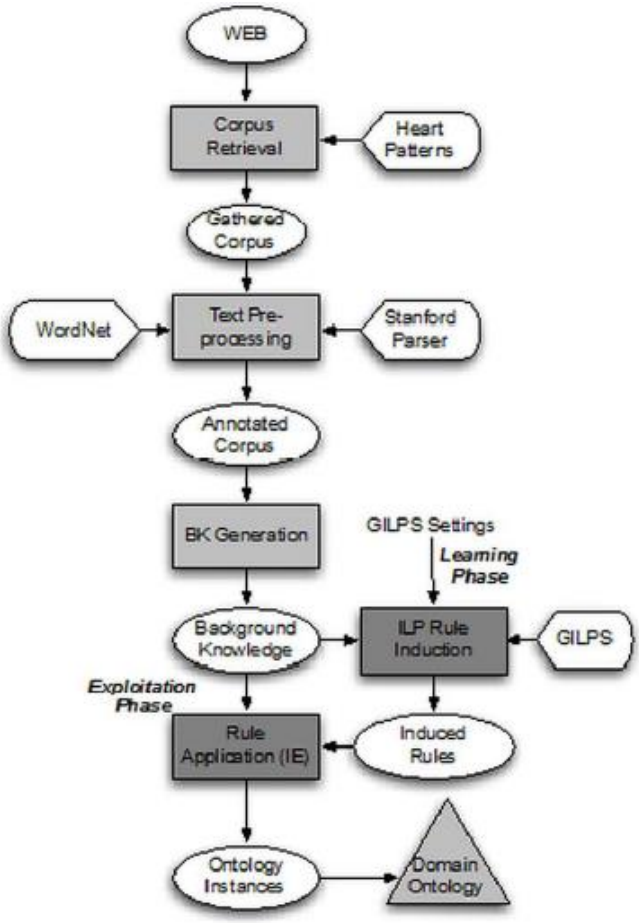


Figura 2-4: Sistema para la población de ontologías basado en PLI. **Fuente:** [14]

Nederstig *et al.* [20] presentan un proceso semi-automático para poblar ontologías, especialmente valores de atributos relacionados con información de productos desde textos semi-estructurados o almacenes web. Inicialmente, se utiliza una ontología predefinida y compatible con la ontología ***GoogRelation***, que se utiliza para el dominio de comercio electrónico. Luego, el método contiene un léxico junto con patrones para clasificar productos, mapear propiedades y crear valores de instancias.

Colace *et al.* [63] presentan un sistema para el aprendizaje y población de ontologías, que combina metodologías estadísticas (*Latent Dirichlet Analysis*, *LDA*) y semánticas (*WordNet*). El sistema recibe como entrada un conjunto de documentos, desde diferentes fuentes web o desde colecciones relacionadas específicas para un dominio de interés y que se clasifican de acuerdo a temas disyuntos semánticamente, para producir una ontología terminológicamente. Incluye dos componentes principales: (i) aprendizaje de

ontologías, que usa *LDA* sobre los documentos de entrada y produce una representación *WWP* (*Weighted Word Pairs*), que contiene los conceptos del dominio más relevantes y sus valores de co-ocurrencia (relaciones) en el conjunto que se analiza; (ii) refinamiento de ontologías, que utiliza propósito general o bases de datos léxicas de dominio específico, refina los conceptos descubiertos previamente, explota sus relaciones léxicas (por ejemplo, relaciones taxonómicas is_a), agrega conceptos ocultos y produce el esquema de la ontología final y la población de la misma. Algunos experimentos se llevan a cabo con documentos TREC-8 para demostrar la efectividad del enfoque propuesto.

Santos y Girardi [64] presentan Apponto-Pro, que es la unificación de varias propuestas. Proponen un proceso incremental (véase la **Figura 2-5**), para lograr la construcción y posterior población de una ontología de aplicación. El sistema es capaz de generar todos los elementos de la ontología tales como clases, taxonomía, relaciones no taxonómicas, instancias, propiedades y axiomas en un archivo de extensión *OWL*. El proceso se compone de seis fases; (i) recolección de objetivos, que requiere que un experto entregue al sistema un conjunto de objetivos, los cuales se utilizan para las siguientes fases; (ii) construcción de una ontología base, que tiene como entrada un conjunto de objetivos que el usuario alimenta manualmente y entrega como salida una ontología base con clases, taxonomía, propiedades y axiomas; (iii) aprendizaje de clases y relaciones taxonómicas, que aprende otras clases y relaciones taxonómicas mediante un algoritmo que extrae elementos a partir de un corpus y la ontología base; (iv) aprendizaje de relaciones no taxonómicas, en el cual se aplican técnicas estadísticas y de procesamiento de lenguaje natural y se realizan otras actividades como la anotación del corpus; (v) población de ontologías, en la cual se realiza la identificación, extracción y clasificación de instancias de relaciones no taxonómicas y propiedades de una ontología desde el corpus anotado y se utilizan técnicas de procesamiento de lenguaje natural y extracción de información para obtener como salida la ontología poblada; (vi) inserción de axiomas, en la cual se necesita la ontología poblada de la fase anterior y utiliza reglas de inferencia en programación lógica inductiva, para lograr extraer nuevas reglas en lógica de primer orden para nuevas relaciones e instancias. Finalmente, se entrega una ontología totalmente poblada y con nuevas reglas. Las pruebas se realizan bajo el dominio derecho de familia.

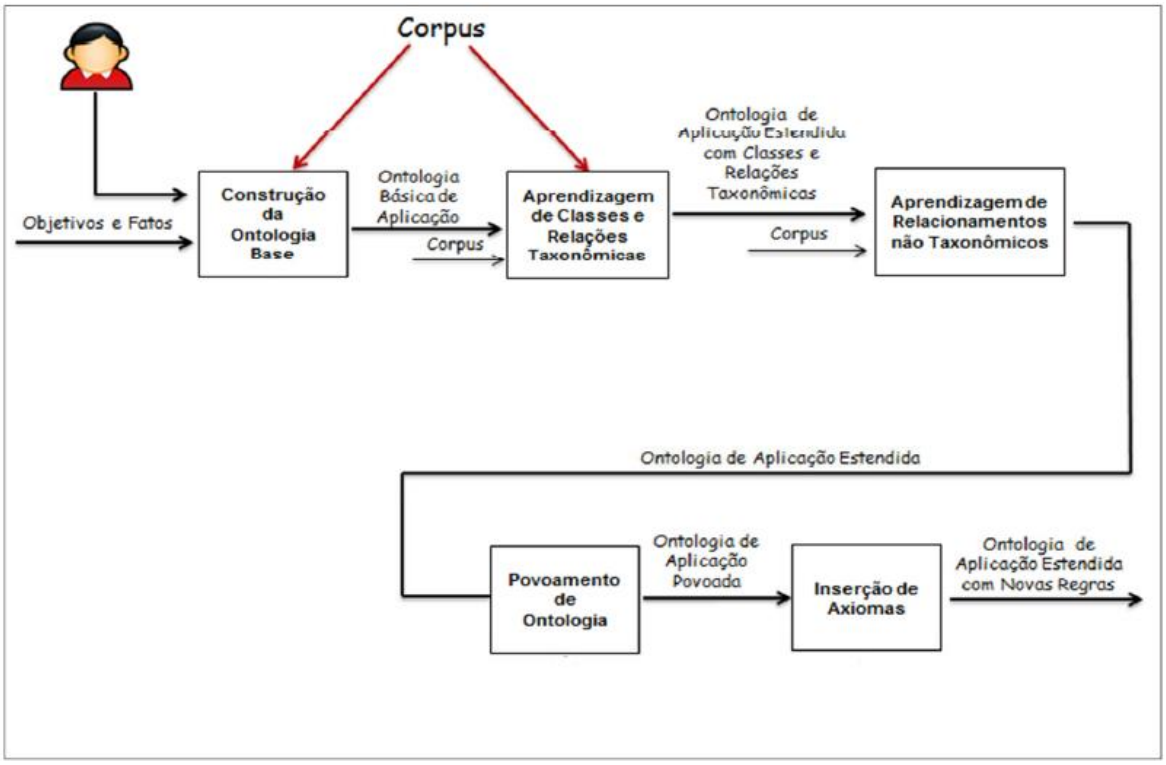


Figura 2-5: Visión general del proceso Apponto-Pro. **Fuente:** Adaptado de [64]

Kordjamshidi y Moens [10] presentan un *framework* para poblar instancias de relaciones en ontologías desde el lenguaje natural que contiene un modelo estructurado de aprendizaje de máquina, el cual tiene muchas variables y restricciones. Una estrategia que se utiliza es subdividir el problema en subproblemas. A su vez, cada subproblema se resuelve por medio de la programación lineal. Se utilizan conceptos de relaciones espaciales tales como trayectoria, puntos de referencia, e indicadores espaciales. Para las pruebas del *framework* utilizan datos de los métodos de evaluación SemEval-2012 y SemEval-2013.

En general, la revisión evidencia algunos problemas en el proceso de población de ontologías. Los métodos deben ser independientes del dominio de aplicación [13,14]. Debido a que el proceso de población de ontologías es bastante costoso, es necesario realizarlo mediante la utilización de métodos semi-automáticos o automáticos [15–17]. La extracción de información para poblar ontologías se debe realizar desde cualquier fuente de información [18,19]. Finalmente, se requiere que los métodos cuenten con buenos niveles de *precision*, *recall* y por consiguiente de *F-measure* [12,20].

Para sintetizar los diferentes enfoques encontrados en la revisión de literatura, se utiliza la Tabla 2-1 con los criterios más importantes. Para ello se tendrá como base la revisión sistemática de literatura realizada por Kitchenham en [65]. Las preguntas orientadoras de la revisión de literatura son:

¿Cuál es el nivel de actualidad de los documentos revisados para la población de ontologías?

¿De qué tipo de documentos (Estructurados, Semi-Estructurados o Libre) los autores realizan la población de ontologías?

¿Cuáles son los dominios (General/Específico) tenidos en cuenta para realizar la población de ontologías?

¿Qué tipo de técnicas utilizan para realizar la población de ontologías?

¿Cuál es el nivel de automatización del método desarrollado?

¿Cuál es el nivel de *precision* y *recall* del método evaluado?

¿El método evaluado construye una ontología desde cero?

Tabla 2-1: Síntesis de los trabajos sobre extracción de instancias de una clase. **Fuente:** Elaboración propia

Criterios / Autores	Tipo de texto procesado	Dominio General/Específico	Técnicas usadas	Nivel de Automatización	Precision (%)	Recall (%)	¿Genera Ontología?
Contreras 2004 [15]	Libre	General (Relaciones Geopolíticas)	Natural language processing y otras	Automático	--	--	Si
Pasca 2004 [18]	Semi-Estructurado y no estructurado	General (5 Categorías)	Patrones Léxico-Sintácticos	Semi-Automático	88,0%	--	No
Geleijnse and Korst 2005 [16]	Semi-Estructurado	General (Cine)	Patrones *Hearst*	Semi-Automático	78,0%	93,8%	No
De Boer *et al.* 2007 [54]	Semi-Estructurado	Específico (Fútbol y patrimonio cultural)	Co-ocurrencia en la web	Semi-Automático	--	--	No
Yoon *et al.* 2007 [55]	Libre	Específico (Dispositivos electrónicos)	*Natural language processing*	Semi-Automático	95,2%	--	No
Talukdar *et al.* 2008 [11]	Estructurado y No Estructurado	General (5 Clases)	Algoritmo ADSORPTION	Semi-Automático	77,4%	--	No
Manine *et al. 2008* [56]	Semi-Estructurado y Estructurado	Específico (Biomédico)	*Information Extraction, Machine Learning*	Automático	89,6%	89,3%	No
Ruiz-Martínez *et al.* 2008 [57]	Semi-estructurado y Libre	Específico (Turismo)	Procesamiento de lenguaje natural.	Automático	93,2%	94,9%	No
Danger y Berlanga 2009 [58]	Semi-Estructurado	General (Arqueología)	Proceso no monolítico	Automático	90,0%	90,0%	No

Tabla 2-1: Síntesis de los trabajos sobre extracción de instancias de una clase. **Fuente:** Elaboración propia

Criterios / Autores	Tipo de texto procesado	Dominio General/Específico	Técnicas usadas	Nivel de Automatización	Precision (%)	Recall (%)	¿Genera Ontología?
Faria y Girardi 2011 [49]	Libre	Específico (Derecho de Familia)	*Natural language processing y Information Extraction*	Semi-Automático	95,0%	75,0%	No
Schlaf y Remus 2012 [59]	Libre	Específico (4 Categorías)	*Machine Learning*	Automático	84,9%	45,0%	No
IJntema *et al.* 2012 [19]	Libre	Específico (Finanzas y Política)	Patrones Léxico-Sintácticos, Léxico Semánticos	Semi-Automático	80,0%	70,0%	No
Ruiz-Martinez *et al.* 2012 [60]	Libre	Específico (Biomédico)	Procesamiento de lenguaje natural.	Automático	79,6%	69,0%	No
Faria *et al.* 2012 [17]	Libre	General (Turismo y Legal)	*Natural language processing y Information Extraction*	Automático	81,9%	82,0%	No
De Araujo *et al.* 2013 [12]	Libre	Específico (Legal)	*Information Extraction, Natural Language Processing*	Automático	98,0%	91,5%	No
Sadoun *et al. 2013* [61]	Libre	Específico (Espacios inteligentes)	*Machine Learning y Reglas de Extracción*	Automático	95,0%	63,0%	No

Tabla 2-1: Síntesis de los trabajos sobre extracción de instancias de una clase. **Fuente:** Elaboración propia

Criterios / Autores	Tipo de texto procesado	Dominio General/Específico	Técnicas usadas	Nivel de Automatización	Precision (%)	Recall (%)	¿Genera Ontología?
Ríos 2013 [62]	Libre	Específico (Turismo)	Procesamiento del lenguaje natural, algoritmo de agrupamiento, Extracción de información	Automático	56,8%	--	Si
Faria *et al. 2013* [13]	Libre	General (Turismo y Legal)	Natural language processing y Information Extraction	Automático	87,3%	86,5%	No
Lima *et al.* 2014 [14]	Semi - Estructurado	General (Nombres de ciudades, enfermedades, aves, pescados y mamiferos)	Programación Lógica Inductiva, *Machine Learning, Natural Language Processing*	Automático	94,0%	59,0%	No
Nederstig *et al.* 2014 [20]	Semi-Estructurado	Específico (E-Commerce)	Léxico y Patrones	Semi-Automático	96,0%	89,0%	No
Colace *et al.* 2014 [63]	Libre	General	Metodologías estadísticas y semánticas	Automático	--	--	Si
Santos y Girardi 2014 [64]	Libre	Específico (Derecho de Familia)	Ciclo Incremental por Objetivos	Semi-Automático	--	--	Si
Kordjamshidi y Moens 2015 [10]	Libre	General	*Machine Learning*	Automático	--	--	No

3. Planteamiento del Problema

3.1 Problema

Las ontologías son una tecnología importante porque brinda las herramientas que permiten estructurar la información de tal forma que, al escribir algunos axiomas, se puede inferir nuevo conocimiento. Para el sector productivo esto significará mayor productividad, debido a que tendrán mayor capacidad de extraer información importante sobre los clientes y sus preferencias.

Las empresas podrán trabajar sobre una misma estructura de información todo lo referente al negocio, pero guardando siempre la privacidad de sus datos. Por ejemplo, si un agente busca una pizza que satisfaga las preferencias de algún cliente, puede utilizar las ontologías existentes sobre pizzas para hacer su elección, después empleará las ontologías empresariales para hacer su pedido.

Es necesario en primera medida la construcción de ontologías bien definidas, por otro lado, se requiere que esas ontologías contengan información actual y allí es donde toma fuerza la población de ontologías desde cualquier fuente de información, preferiblemente desde documentos web, por ello es necesario la construcción de métodos semi-automáticos o automáticos que ayuden en todo el proceso de construcción y población de ontologías. Un método de población de ontologías habrá logrado las metas siempre y cuando logre obtener buenos niveles de *precision*, *recall* y *F-measure*.

La revisión de literatura muestra que, aunque la mayoría de métodos de población de ontologías o extracción de instancias de una clase que se desarrollaron hasta ahora, arrojan buenos resultados en comparación con su promedio de precisión (*precision*) y algunos en su nivel de exhaustividad (*recall*), algunos de estos métodos no consideran y la extracción de instancias desde un texto en lenguaje natural sin tener la limitación del

dominio y otros sólo la consideran parcialmente. Otros métodos realizan el proceso desde textos semi-estructurados o estructurados.

En algunos casos los dominios en que trabajan los métodos son bastante limitados. De Boer *et al* [54] obtienen un 87% de las instancias posibles desde textos semi-estructurados, pero trabajan solamente los dominios de patrimonio cultural y fútbol. Faria y Girardi [49] presentan un método para semi-automatizar la población de ontologías desde textos y obtienen un 95% de *precision*, pero solamente trabajan el dominio derecho de familia. Schlaf y Remus [59] extraen instancias únicamente de cuatro categorías: nombre, apellido, profesión y ciudad. Ruiz-Martínez *et al.* [60] presentan un método para población de ontologías que recibe cualquier tipo de texto, lo procesa de forma automática y obtienen un 79% de *precision* y 69% de *recall*; el método trabaja el dominio biomédico. Sadoun *et al.* [61] presentan un método que realiza la extracción del 95% de las instancias posibles desde un texto sin estructura, pero solamente trabajan el dominio de espacios inteligentes y reconocen que falta trabajar otros dominios. Ríos [62] propone un método para la población de ontologías, que utiliza textos de información en cualquier formato en el dominio de turismo.

En otros proyectos mencionan que la extracción de instancias se puede realizar desde cualquier dominio [11,13,14,18]. El problema de muchos de estos métodos es que se deben alimentar con semillas (en algunos casos instancias y clases semillas). En otros, los sistemas se deben alimentar con reglas para poder realizar la extracción. Si bien en estos trabajos se sugiere que la aplicación se puede realizar en otros dominios, las instancias, clases semilla y reglas se atan casi completamente a un dominio específico. Otro problema de estos métodos es que las instancias que encuentran son nombres de personas, nombres de compañías, etc. Las instancias de nombres propios tienen una dificultad menor, pues se podrían extraer, incluso, desde listas predefinidas. Existen otros tipos de instancias de mayor dificultad y que no se tienen en cuenta. Finalmente, muy pocos de los proyectos revisados apuntan a la población de instancias de atributos, lo cual hace que los porcentajes de *recall* sean inferiores a los que realmente están obteniendo [13,19,62].

Según la literatura consultada la población de ontologías es un campo de gran importancia para la comunidad científica en la actualidad. Por esta razón, en esta Tesis

Doctoral se busca avanzar en la extracción de instancias de una clase o población de ontologías, realizando dicha extracción desde textos escritos en lenguaje natural y teniendo en cuenta la independencia del dominio de aplicación.

3.2 Objetivos

3.2.1 Objetivo General

Desarrollar un método para la identificación y extracción de instancias de una clase desde lenguaje natural independientes del dominio de aplicación.

3.2.2 Objetivos específicos

- Explorar los diferentes conceptos relacionados con la población de ontologías y los diferentes enfoques que se emplean en su realización.
- Revisar las diferentes técnicas que pueden abordar el problema planteado y examinar su uso potencial para solucionarlo.
- Diseñar un método para aplicarlo al problema planteado.
- Realizar una revisión de las posibles herramientas de desarrollo y, de acuerdo con esto, realizar una implementación bajo un ambiente de programación.
- Evaluar el método propuesto mediante las pruebas en el desarrollo y bajo los criterios de nivel de *recall* y *precision*.

3.3 Hipótesis

Es posible desarrollar un método computacional que logre identificar y extraer las instancias de una clase a partir del lenguaje natural y sin importar el dominio de aplicación, utilizando técnicas como extracción de información, procesamiento de lenguaje natural y ontologías.

3.4 Metodología

Con base al Framework Design Science [66] y a los objetivos planteados del proyecto de investigación se siguió entonces, una metodología basada en cuatro fases, a saber:

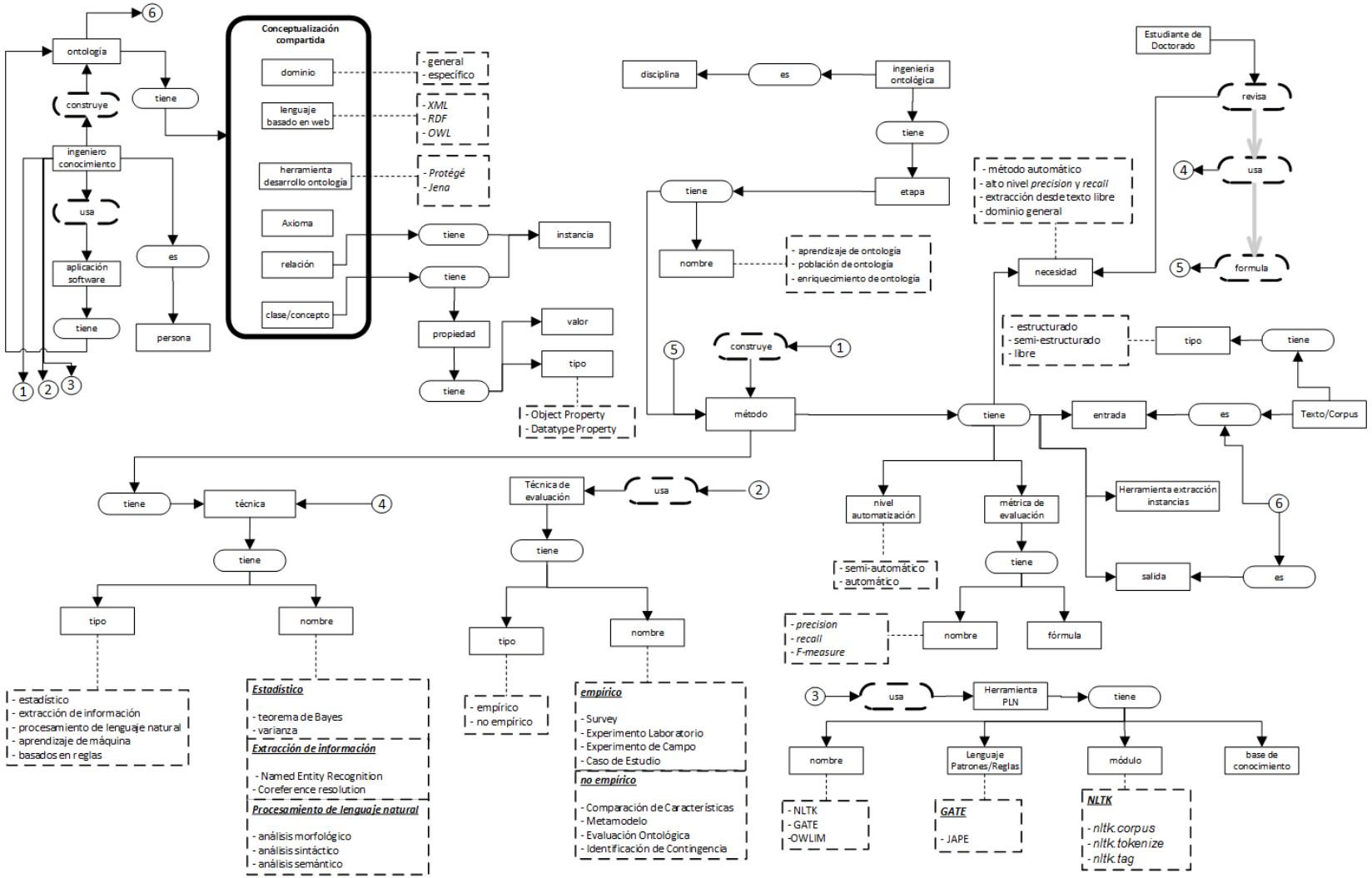
- Exploración: Donde se realizó la Revisión Documental de la teoría sobre ontologías y de los métodos existentes para población de ontologías.
- Síntesis: Donde se tomaron los elementos conceptuales de la revisión para conceptualizar el diseño del método.
- Desarrollo: Donde se desarrolló la solución que se conceptualizó en la fase anterior y se implementó en la herramienta computacional GATE, la cual permitió realizar pruebas al método y ver su desempeño en varios dominios.
- Validación: Donde se realizó la evaluación del funcionamiento de la solución utilizando la técnica de evaluación de métodos comparación de características y teniendo como base los criterios *precision, recall* y *F-measure*.

El método propuesto se analizó y evaluó bajo los criterios mencionados. Seguidamente se creó un documento, donde se midieron los resultados del proyecto, esto fue la base teórica para aportar ideas de nuevos proyectos que se deseen emprender.

4. Solución

En esta Tesis Doctoral se propone diseñar e implementar un método computacional automático que permita desde un texto escrito en lenguaje natural, construir y poblar una ontología y luego generar un archivo OWL. En este Capítulo se presenta el método propuesto, el cual utiliza técnicas de extracción de información y procesamiento de lenguaje natural para lograr la población automática de ontologías mediante la extracción de instancias desde textos en lenguaje natural y con independencia del dominio de aplicación. En la **Figura 4-1** se presenta el contexto de la aplicación del método en la población automática de ontologías, para ello se utiliza un esquema preconceptual [67]. Allí se trata de describir las entradas y salidas, los actores humanos y los roles específicos dentro del marco del método creado. También se presentan los principales conceptos teóricos y cómo se conectan con el problema que se plantea y la solución a la que se llega utilizando las herramientas existentes encontradas en la revisión de literatura.

Figura 4-1: Contexto de aplicación del método en la población de Ontologías. **Fuente:** Elaboración Propia



En la **Figura 4-2** se presenta una visión general del método que se desarrolla a lo largo del Capítulo.

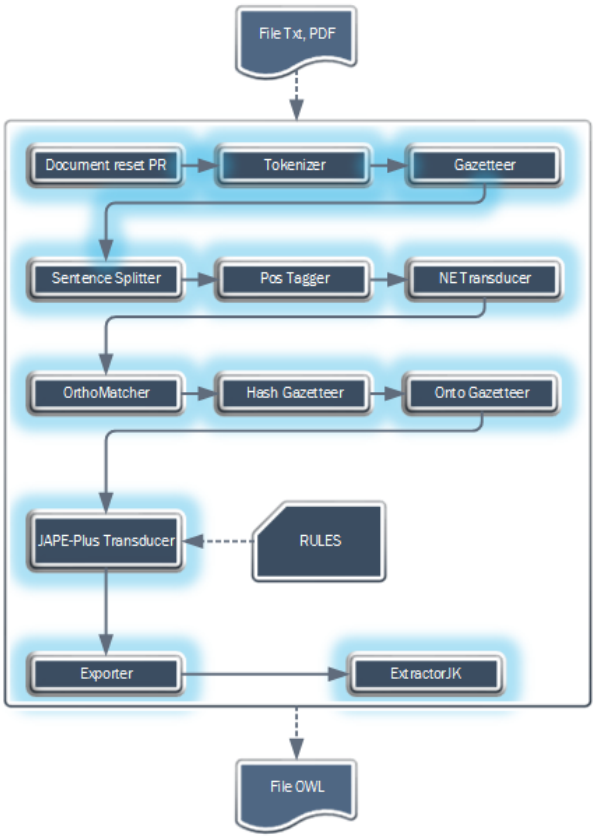


Figura 4-2: Visión general del proceso de población automática de Ontologías. **Fuente:** Elaboración Propia

El ambiente de programación que se utiliza para el desarrollo de la solución al problema planteado es GATE (*General Architecture for Text Engineering*) [21]. Este lenguaje se utiliza para implementar modelos que permiten resolver problemas concernientes al procesamiento de lenguaje natural.

GATE permite definir la organización de un sistema y tiene un conjunto de módulos que se pueden reusar, extender o adaptar, de modo que ayuda a disminuir el tiempo de desarrollo. El ambiente visual permite que se puedan desarrollar aplicaciones pensadas para la integración de módulos, lo cual es un gran aporte al concepto de reúso que se maneja en ingeniería de software. También se puede utilizar desde Java como una API, lo que permite que se puedan crear aplicaciones un poco más sofisticadas de escritorio, en ambiente web o incluso dispositivos móviles [51].

4.1 Archivo de Entrada

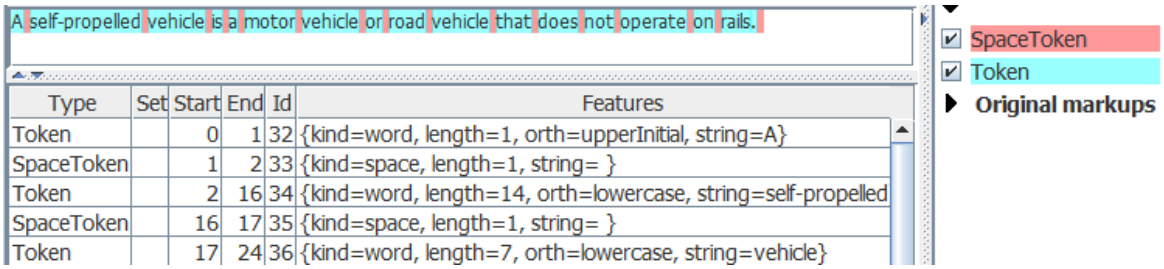
Existen diferentes tipos de fuentes de información: estructuradas, que se refieren a la información que se encuentra contenida en bases de datos relacionales; semi-estructuradas, que regularmente se alojan en páginas web en formato HTML; basadas en lenguaje natural, que no tienen ningún tipo de procesamiento más que la redacción de las personas que las escriben. La entrada del método propuesto es un archivo en lenguaje natural y con extensión txt o pdf, que es el único recurso que se solicita al usuario y en el cual se basa el análisis posterior, para realizar la extracción de instancias de una clase.

4.2 Proceso *Document Reset* PR (*Processing Resource*)

Este proceso consiste en reiniciar el documento (corpus), es decir, eliminar anotaciones que pueden alterar el proceso que se piensa iniciar, dado que el texto se procesara anteriormente con algún recurso de GATE. Es válido aclarar que allí existen varias opciones, las cuales permiten definir las necesidades del procesamiento en determinado momento. En otras palabras, se restaura el documento a su estado inicial sin ningún tipo de anotaciones dentro de su contenido.

4.3 Proceso *Tokenizer*

Es una técnica que se basa en procesamiento de lenguaje natural mediante la utilización de reglas JAPE. Aquí se realiza el proceso de separar las palabras que se encuentran en el texto en simples *tokens*. Se pueden encontrar viarios tipos de *tokens,* los cuales tienen atributos de cadenas y un tamaño. Los *tokens* pueden ser palabras, números, símbolos, signos de puntuación y *tokens* de espacio en blanco o saltos de línea. En la **Figura 4-3**, se presenta una oración en lenguaje natural con sus respectivos *tokens*. La parte azul indica que cada una de las palabras contiene un *token*; asimismo, se muestran los *SpaceToken* marcados con un color rosado. Por ejemplo, el tercer *token* es la palabra "*self-propelled*", que se encuentra en minúscula, tiene una longitud de 14 y es de tipo palabra. De la misma forma, la segunda unidad lingüística es un *SpaceToken* de tamaño uno y corresponde a un espacio en blanco. Este análisis se realiza para cada una de las palabras contenidas en el texto.



Type	Set	Start	End	Id	Features
Token		0	1	32	{kind=word, length=1, orth=upperInitial, string=A}
SpaceToken		1	2	33	{kind=space, length=1, string= }
Token		2	16	34	{kind=word, length=14, orth=lowercase, string=self-propelled
SpaceToken		16	17	35	{kind=space, length=1, string= }
Token		17	24	36	{kind=word, length=7, orth=lowercase, string=vehicle}

Figura 4-3: Proceso *Tokenizer*. **Fuente:** Elaboración Propia

4.4 Proceso *Gazetteer*

En este proceso se identifican nombres de entidades teniendo como base una serie de listas predefinidas. Las listas son archivos planos con una entrada por línea que se compilan mediante máquinas de estados finitos y se utilizan para darle algún tipo de memoria al sistema. Las listas se enriquecen y cuentan con nombres de lugares, organizaciones, nombres de personas, entre otros. También se define un archivo con extensión "**.def",* el cual permite tener acceso a todas las listas predefinidas, que a su vez deben tener definido un tipo *MajorType* y opcionalmente un *MinorType*. Por ejemplo, al definir una lista llamada "*city.lst"* que hace referencia a una lista de ciudades de todo el mundo, esta lista tiene un tipo "*MajorType*", que se refiere a que es un lugar, y un tipo "*MinorType*" que hace referencia a que es una ciudad (también podría ser país). Estos tipos se deben especificar al construir la regla en lenguaje JAPE y se usan de acuerdo con la necesidad de extracción. Las listas predefinidas se pueden enriquecer de forma manual, se debe ingresar a la carpeta *Gazetteer* de la carpeta raíz de *GATE*, luego se debe escoger el archivo de la lista que se desea enriquecer y se agregan las palabras al archivo, después se guardan los cambios. Las listas contienen palabras de muchos dominios diferentes y se utilizan como un proceso adicional para encontrar otras entidades y esto no afecta la independencia del dominio del sistema. En la **Figura 4-4** se presenta una oración en lenguaje natural que luego se procesa con el sistema propuesto utilizando la regla "*Lookup.minorType==city*", la cual define que se deben extraer los nombres de ciudades que se encuentran en el texto y en la lista invocada. Se obtiene como resultado que "Paris" es de tipo "*majorType location*" y "*minorType city*". Finalmente, se etiqueta "Paris" como "*Lookup*" debido a que se encuentra en una de las listas predefinidas, para este ejemplo en la lista llamada "*city.lst"*.

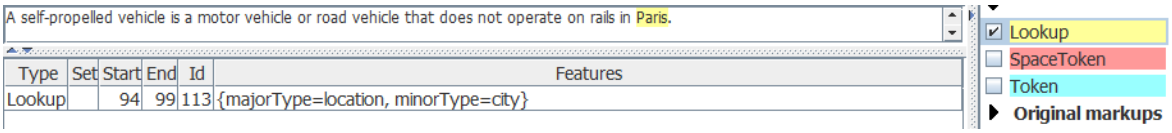


Figura 4-4: Proceso *Gazetteer*. **Fuente:** Elaboración Propia

4.5 Proceso *Sentence Splitter*

Este proceso divide el texto en oraciones, para lo cual se utilizan transductores de estado finito, es decir, alfabetos de entrada y salida. Se utilizan listas de abreviaturas para distinguir puntos aparte. En la **Figura 4-5** se presenta una oración en lenguaje natural. Se puede evidenciar que el sistema identifica que la oración es la que termina en el punto aparte (color verde) y la etiqueta como "*Sentence*"; también, se reconoce el punto aparte y lo etiqueta como "*Split*".

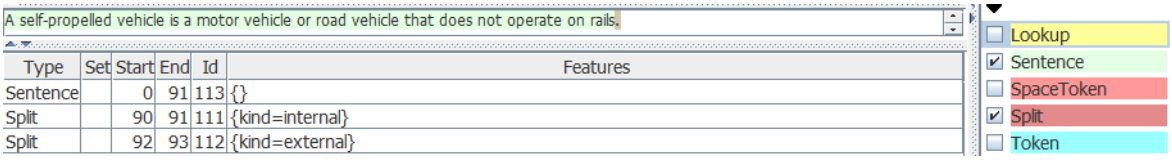
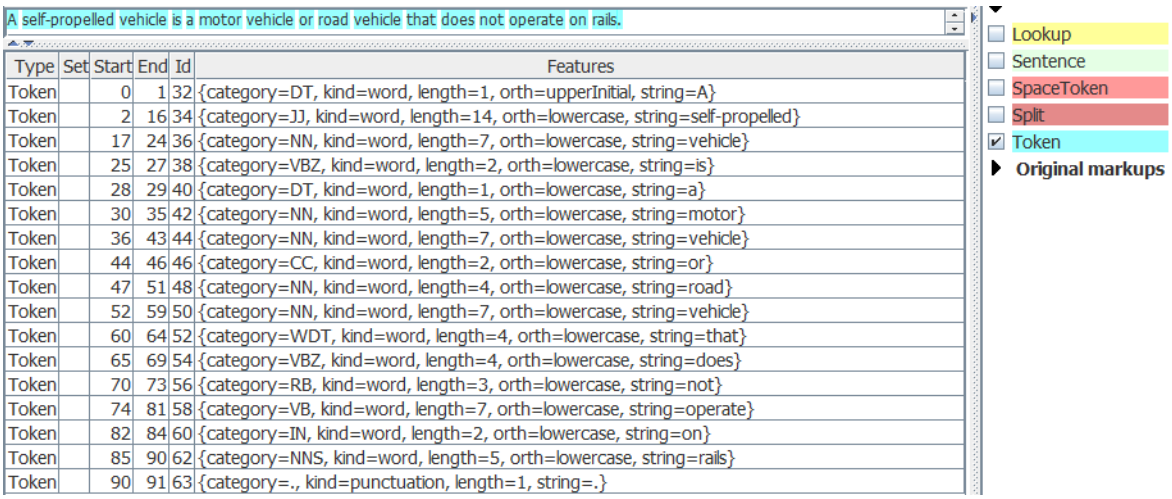


Figura 4-5: Proceso *Sentence Splitter*. **Fuente:** Elaboración Propia

4.6 Proceso *PosTagger*

El proceso utiliza un lexicón y un conjunto de reglas en lenguaje JAPE para agregar categorías gramaticales a cada palabra que se encuentra en el texto, mediante anotaciones *Penn Treebank,* que son estándar para el idioma inglés [68]. En la **Figura 4-6** se presenta una oración en lenguaje natural. Después de aplicar el *PosTagger,* a cada una de las palabras se le asigna una categoría y la oración queda de la siguiente forma: ***A***(DT-determinante) ***self-propelled***(JJ-adjetivo) ***vehicle***(NN-sustantivo) ***is***(VBZ–verbo en tercera persona singular presente) ***a***(DT-determinante) ***motor***(NN-sustantivo) ***vehicle***(NN-sustantivo) ***or***(CC-conjunción) ***road***(NN-sustantivo) ***vehicle****(NN-sustantivo)* ***that***(WDT–determinante WH) ***does***(VBZ–verbo tercera persona singular presente) ***not***(RB-adverbio) ***operate***(VB–verbo en forma base) ***on***(IN–preposición o conjunción subordinada) ***rails***(NNS–sustantivo plural). Este análisis también se realiza para todas las palabras que conforman el texto.



Type	Set	Start	End	Id	Features
Token		0	1	32	{category=DT, kind=word, length=1, orth=upperInitial, string=A}
Token		2	16	34	{category=JJ, kind=word, length=14, orth=lowercase, string=self-propelled}
Token		17	24	36	{category=NN, kind=word, length=7, orth=lowercase, string=vehicle}
Token		25	27	38	{category=VBZ, kind=word, length=2, orth=lowercase, string=is}
Token		28	29	40	{category=DT, kind=word, length=1, orth=lowercase, string=a}
Token		30	35	42	{category=NN, kind=word, length=5, orth=lowercase, string=motor}
Token		36	43	44	{category=NN, kind=word, length=7, orth=lowercase, string=vehicle}
Token		44	46	46	{category=CC, kind=word, length=2, orth=lowercase, string=or}
Token		47	51	48	{category=NN, kind=word, length=4, orth=lowercase, string=road}
Token		52	59	50	{category=NN, kind=word, length=7, orth=lowercase, string=vehicle}
Token		60	64	52	{category=WDT, kind=word, length=4, orth=lowercase, string=that}
Token		65	69	54	{category=VBZ, kind=word, length=4, orth=lowercase, string=does}
Token		70	73	56	{category=RB, kind=word, length=3, orth=lowercase, string=not}
Token		74	81	58	{category=VB, kind=word, length=7, orth=lowercase, string=operate}
Token		82	84	60	{category=IN, kind=word, length=2, orth=lowercase, string=on}
Token		85	90	62	{category=NNS, kind=word, length=5, orth=lowercase, string=rails}
Token		90	91	63	{category=., kind=punctuation, length=1, string=.}

Figura 4-6: Proceso *PossTagger*. **Fuente:** Elaboración Propia

4.7 Proceso NE *Transducer*

Este proceso se basa en técnicas de extracción de información y también se conoce como etiquetado semántico. Recibe como parámetro un archivo “main.jape”, el cual contiene una lista de gramáticas en lenguaje JAPE. De esta forma se logra etiquetar entidades que se refieran a compañías, lugares, fechas y monedas, entre otros. La diferencia con los *Gazetteer* radica en que, aunque algunas gramáticas usan las listas predefinidas, el proceso no depende totalmente de ellas debido a las gramáticas que se especifican en lenguaje JAPE. En la **Figura 4-7** se presenta un párrafo en lenguaje natural, donde se extraen varias entidades como los nombres “*Brown*” y “*Simpson*”, las fechas y, finalmente, un valor de “*$40 million*” que pertenece a la etiqueta moneda.

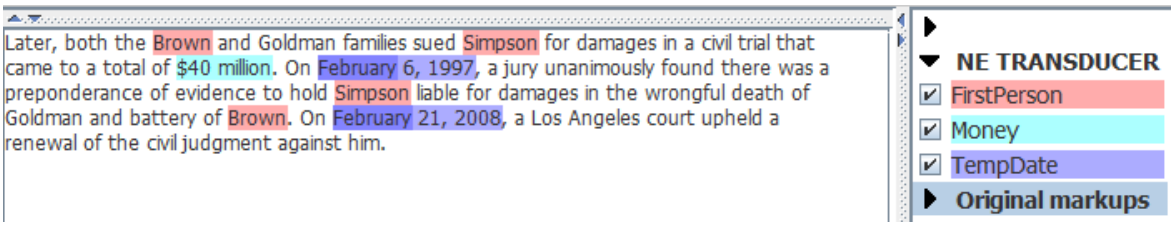


Figura 4-7: Proceso NE *Transducer*. **Fuente:** Elaboración Propia

4.8 Proceso *OrthoMatcher*

En este proceso se usa una técnica de extracción de información. Un proceso de correferencia ocurre cuando, dentro de un texto, existen dos o más expresiones que

hacen referencia a un mismo texto o cosa. Por ejemplo, en la oración “el automóvil de Camilo, donde él viaja mucho”, se puede observar que “él”, realmente hace referencia a la entidad nombrada “Camilo”, que el sistema ya debió etiquetar. Esto fortalece el sistema porque se confirman muchas de las entidades y se pueden encontrar otras. En la **Figura 4-8** se presenta un párrafo donde se encuentran dos entidades utilizando correferencia. La primera es “*Brown*” donde, gracias a las reglas, el sistema detecta que al nombrar por segunda vez la palabra “*Brown*” después de la palabra “*of*”, realmente se refiere a la primera entidad que antecede un determinante. En el segundo caso, Simpson se detecta gracias a una regla que define que, si después del verbo sigue un nombre propio y esta estructura se vuelve a presentar, existe correferencia si los nombres son iguales, es decir, se habla de la misma entidad.

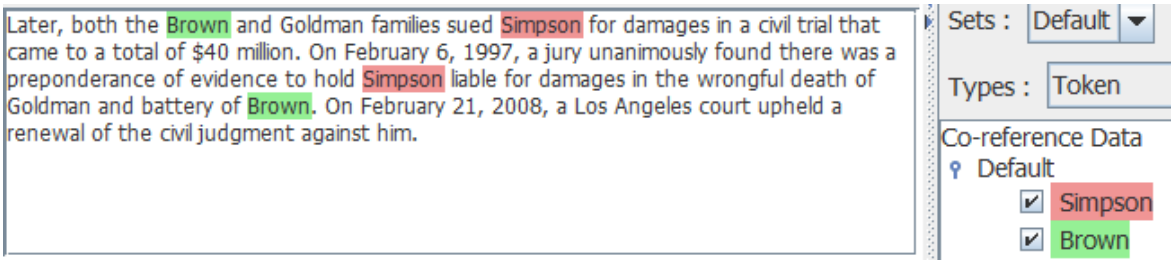


Figura 4-8: Proceso de *Co-reference*. **Fuente:** Elaboración Propia

4.9 Proceso *HashGazetteer*

Este proceso se utiliza como un refinamiento y es adicional, significa que no afecta la independencia del dominio del sistema, pues se buscan otras entidades por medio de listas predefinidas de GATE mediante un algoritmo de búsqueda, que se basa en tablas *Hash* de palabras. El proceso se debe alimentar con una cadena de entrada; luego, se divide en varias partes, para lo cual se utiliza el espacio en blanco como delimitador. Por ejemplo, la oración “el perro es grande” tiene tres partes: “perro” es la primera, “es” la segunda y “grande” la tercera. También, existe una lista de *HashMaps*, la cual se define teniendo en cuenta las partes encontradas: la primera parte va en el primer mapa, la primera parte más un espacio en blanco más la segunda parte van en el segundo mapa y así sucesivamente. Al final, la oración completa se coloca en el mapa apropiado y se referencia el objeto “*Lookup*”, que se encuentra junto a él. Con ello se busca confirmar las entidades y encontrar nuevas [51].

4.10 Proceso *OntoGazetteer*

Este proceso también se utiliza para encontrar entidades en el texto, pero es necesario utilizar las listas predefinidas de GATE. La diferencia con los otros "*Gazetteer*", radica en que permite vincular anotaciones a las clases de una ontología. Para realizar el proceso, se debe contar con uno o varios archivos que contengan listas con extensión "*.lst". Se requieren dos archivos de definición: (i) el primero es un archivo con extensión "*def*", que define las relaciones entre los archivos de la lista y los conceptos de la ontología; (ii) el segundo contiene las relaciones entre las listas y las anotaciones que se generan. Estas anotaciones tienen el mismo tipo "*Lookup*", igual que con los otros *Gazetteer* [51]. Este proceso es importante para esta Tesis Doctoral porque permite encontrar otras entidades. La vinculación a entidades ontológicas no se realiza aquí sino más adelante en el proceso *ExtractorJK*.

4.11 Proceso JAPE-Plus *Transducer-Rules*

Un sistema para poblar ontologías forzosamente debe recibir una ontología vacía la cual, después de un proceso de extracción de información desde textos en lenguaje natural, se debe poblar completamente con instancias de conceptos y relaciones de esa ontología. En esta Tesis Doctoral se propone realizar un proceso completamente automático y, para tal fin, se decide diseñar e implementar un sistema que sea capaz de extraer diferentes entidades ontológicas, tales como clases, subclases, instancias, atributos de clases, valores de atributos y relaciones. Para lograrlo, se diseña un sistema de reglas con patrones sintácticos que sirven de apoyo para etiquetar las diferentes entidades, que constituye uno de los mayores aportes del trabajo. JAPE-*Plus Transducer-Rules*, es un *plugin* de GATE, este permite ejecutar el sistema de reglas que fueron implementadas en lenguaje JAPE.

A continuación, se presenta el sistema de reglas que se utiliza para la extracción de clases, instancias, atributos y relaciones. En el cuerpo del trabajo se presentan algunas reglas; las demás se proponen en los anexos.

4.11.1 Reglas para extracción de clases

Para la extracción de clases se crean dos tipos de anotaciones: generales y específicas. Para las anotaciones generales se diseñan reglas que definen un tipo de anotación llamada “NPJK”. Las reglas de tipo NPJK se llaman desde cualquier regla JAPE, siempre y cuando se definan como anotaciones de entrada en el encabezado de la regla nueva.

Reglas Generales (NPJK)

Las reglas generales se diseñan básicamente para identificar nombres de clases compuestas de dos o más palabras. Estas reglas se pueden invocar desde la definición de las reglas específicas. En la **Figura 4-9** se presenta una regla en lenguaje JAPE. La regla especifica que si encuentra tres palabras seguidas que son sustantivos, el patrón se etiqueta como “*nounPhrase”* entonces “*nounPhrase”* se define como un tipo NPJK.

```
Phase:NPJK /*Nombre de la Fase*/
Input: Token /*Tipos de anotaciones de entrada */
Options: control = brill /* método que se debe aplicar en caso de superposición
de reglas*/

Rule:NPJK3 /*Nombre de la regla */
Priority: 40 /*Prioridad de la regla */

/*Parte izquierda de la regla: Sustantivo + Sustantivo + Sustantivo*/
(
({Token.length>1,Token.category==NN}{Token.length>1,Token.category==NN}
{Token.length>1,Token.category==NN})
/*Etiqueta de la regla */
):nounPhrase
-->
/*Parte derecha de la regla: Si se cumple el patrón de lado izquierdo
se etiqueta como NPJK*/
:nounPhrase.NPJK={kind="NPJK",rule=NPJK3}
```

Figura 4-9: Regla 3 para etiquetar tipo NPJK. **Fuente:** Elaboración Propia

En la **Figura 4-10** se presenta un patrón para especificar que, al encontrar tres sustantivos seguidos más un sustantivo plural, se etiqueta todo como un “*nounPhrase”,* entonces “*nounPhrase”* se define como un tipo NPJK.

```
Phase:NPJK /*Nombre de la Fase*/
Input: Token /*Tipos de anotaciones de entrada */
Options: control = brill /* método que se debe aplicar en caso de superposición
de reglas*/

Rule:NPJK6 /*Nombre de la regla */
Priority: 45 /*Prioridad de la regla */

/*Parte izquierda de la regla: Sustantivo + Sustantivo + Sustantivo
plural*/
(
({Token.length>1,Token.category==NN}{Token.length>1,Token.category==NN}
{Token.category==NNS})

/*Etiqueta del patrón */
):nounPhrase
-->
/*Parte derecha de la regla: Si se cumple el patrón de lado izquierdo
se etiqueta como NPJK*/
:nounPhrase.NPJK={kind="NPJK",rule=NPJK6}
```

Figura 4-10: Regla 6 para etiquetar tipo NPJK. **Fuente:** Elaboración Propia

Reglas Específicas

Las reglas específicas se diseñan como reglas genéricas, logrando que se puedan extraer clases desde cualquier texto, sin importar el dominio. Esta generalidad, refuerza las clases que se descubren en las etapas anteriores o encuentra nuevas. En la **Figura 4-11** se presenta una regla en lenguaje JAPE, en la que se define, que al encontrar una palabra que es un sustantivo, se etiqueta como sust1; luego, cuando se encuentra otra palabra que es un verbo en tercera persona singular presente, más una palabra que es un determinante, más una palabra que es un adjetivo, más una palabra que es un sustantivo, este último se etiqueta como “sust2”; finalmente, las palabras etiquetadas como “sust1” y “sust2” se ingresan a la lista de clases.

```
Phase:Test1 /*Nombre de la Fase*/
Input:  Token Lookup /*Tipos de anotaciones de entrada */
Options: control = appelt /* Método que se debe aplicar en caso de superposición de reglas*/

Rule: Clases3 /*Nombre de la regla */
Priority: 25 /*Prioridad de la regla */
(
/*Parte izquierda de la regla*/
({Token.length>1,Token.kind==word,Token.category==NN}):sust1 /* Se etiqueta la primera clase*/
{Token.kind==word,Token.category==VBZ}
{Token.kind==word,Token.category==DT}
{Token.kind==word,Token.category==JJ}
({Token.length>1,Token.kind==word,Token.category==NN}):sust2 /* Se etiqueta la segunda clase*/
/*Etiqueta del patrón completo*/
):inst
-->
/*Parte derecha de la regla: Si se cumple el patrón de lado izquierdo se etiquetan las clases
encontradas*/
:sust1.Clases={rule= "Clases3"},
:sust2.Clases={rule= "Clases3"}
```

Figura 4-11: Regla 3 para etiquetar Clases. **Fuente:** Elaboración Propia

En la **Figura 4-12** se presenta una regla que, al encontrar una palabra que es un determinante, más otra palabra(s) tipo NPJK, el tipo NPJK se etiqueta como “sust1” y luego “sust1” se ingresa a la lista de clases. Otras reglas para extraer clases se presentan en el **Anexo A**.

4.11.2 Reglas para extracción de Instancias

Para la extracción de instancias también se crean dos tipos de anotaciones: generales y específicas. Para las anotaciones generales, se crean reglas que definen un tipo de anotación llamada “NPJKINST”. Las reglas de tipo NPJKINST se pueden llamar desde cualquier regla JAPE, siempre y cuando se definan como anotaciones de entrada en el encabezado de la nueva regla.

```
Phase:Test1 /*Nombre de la Fase*/
Input:  Token Lookup NPJK /*Tipos de anotaciones de entrada */
Options: control = appelt /* Método que se debe aplicar en caso de superposición de reglas*/

Rule: Clases14 /*Nombre de la regla */
Priority: 25 /*Prioridad de la regla */
/*Parte izquierda de la regla: Determinante + NPJK*/
(
{Token.category==DT}
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*Se etiqueta el tipo NPJK, sea una o varias palabras*/
/*Etiqueta del patrón completo, DT + NPJK*/
):inst
-->
/*Parte derecha de la regla: Si se cumple el patrón de lado izquierdo se guarda la clase con algunas
características*/
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust1"); /* Se obtiene el conjunto de elementos
etiquetados con sust1*/
FeatureMap features =Factory.newFeatureMap(); /*Se crean las características*/
for(Annotation ann:as) { /*Ciclo para recorrer todos los elementos encontrados*/
/*Se guarda en la variable instanceName la información del elemento encontrado desde la posición
inicial hasta la final*/
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));/*Se le da un tipo al elemento encontrado*/
features.put("rule","Clases14"); /*Se almacena información de la regla*/
features.put("className",ann.getFeatures().get("class")); /*El nombre de la clase encontrada*/
features.put("representationId",ann.getId()); /*Información del ID asignado al elemento en la
caracterítica representationId*/
features.put("id",ann.getId()); /*Información del ID asignado al elemento*/
features.put("corefchain", null); /*Información sobre correferencia, no se utiliza*/
features.put("instanceName", instanceName); /*El valor de la palabra(s) tal como se ecnontró en el
texto*/
}/*Termina el ciclo*/
outputAS.add(as.firstNode(),as.lastNode(),"Clases",features); /*Todos los elementos encontrados, se
agrupan en la etiqueta Clases*/
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

Figura 4-12: Regla 14 para etiquetar Clases. **Fuente:** Elaboración Propia

Reglas Generales (NPJKINST)

Las reglas generales, básicamente, permiten identificar instancias compuestas, por ejemplo "*Charles Brown*", "*Manchester City*" y "*London Bank*", entre otras. Estas reglas se pueden invocar desde la definición (encabezado) de las reglas específicas en lenguaje JAPE. En la **Figura 4-13** se presenta una regla en lenguaje JAPE, la cual permite especificar que para etiquetar a "*nounPhrase*", es necesario encontrar una palabra que sea un nombre propio singular, más un símbolo ".", más otra palabra es nombre propio singular, entonces "*nounPhrase*" se etiqueta como un tipo NPJKINST.

```
Phase:NPJKINST /*Nombre de la Fase*/
Input: Token /*Tipos de anotaciones de entrada */
Options: control = brill /* Método que se debe aplicar en caso de
superposición de reglas*/

Rule:NPJKINST5 /*Nombre de la regla */
Priority:45 /*Prioridad de la regla */

/*Parte izquierda de la regla: Nombre propio singular + símbolo "." +
Nombre propio singular*/
(
({Token.category==NNP,Token.length==1}{Token.category==".",Token.length==1,
Token.string=="."}{Token.category==NNP})
/*Etiqueta del patrón */
):nounPhrase
-->
/*Parte derecha de la regla: Si se cumple el patrón de lado izquierdo se
etiqueta como NPJKINST*/
:nounPhrase.NPJKINST = {kind="NPJKINST",rule=NPJKINST5}
```

Figura 4-13: Regla 5 para etiquetar tipo NPJKINST. **Fuente:** Elaboración Propia

En la **Figura 4-14** se presenta una regla que permite etiquetar a "*nounPhrase*"; se requiere que se encuentren tres palabras seguidas que sean nombre propio singular más un símbolo ".", más otra palabra que sea nombre propio singular; así, "*nounPhrase*" se etiqueta como un tipo NPJKINST.

```
Phase:NPJKINST /*Nombre de la Fase*/
Input: Token /*Tipos de anotaciones de entrada */
Options: control = brill /* Método que se debe aplicar en caso de
superposición de reglas*/

Rule:NPJKINST7 /*Nombre de la regla */
Priority: 45 /*Prioridad de la regla */

/*Parte izquierda de la regla: Nombre propio singular + Nombre propio
singular + Nombre propio singular + símbolo "." + Nombre propio singular*/
(
({Token.category==NNP}{Token.category==NNP}{Token.category==NNP}{Token.cate
gory==NNP}{Token.category=="."}{Token.category==NNP})

/*Etiqueta del patrón */
):nounPhrase
-->
/*Parte derecha de la regla: Si se cumple el patrón de lado izquierdo se
etiqueta como NPJKINST*/
:nounPhrase.NPJKINST = {kind="NPJKINST",rule=NPJKINST7}
```

Figura 4-14: Regla 7 para etiquetar tipo NPJKINST. **Fuente:** Elaboración Propia

Reglas Específicas

Las reglas específicas se diseñan de tal forma que sean reglas genéricas y que puedan servir para extraer instancias en cualquier texto sin importar el dominio. Si se cumple el patrón, entonces la acción consiste en enviar ese valor de atributo con el nombre de atributo "*has_*****" al siguiente proceso del sistema. La creación de la instancia como tal, se realiza en un proceso posterior.

En la **Figura 4-15** se presenta una regla para extraer direcciones de correos electrónicos. En el patrón se especifica la siguiente estructura: una palabra o número una o varias veces, más un símbolo "_" cero o una vez, más un símbolo "." cero o una vez, más una palabra número o símbolo "_" cero o varias veces, más el símbolo "@", más una palabra, símbolo, signo de puntuación o número, más el símbolo "." cero o una vez, más una palabra, símbolo, signo de puntuación o número cero o varias veces, más el símbolo "." cero o una vez, más una palabra, símbolo, signo de puntuación o número cero o una vez, más el símbolo "." cero o una vez, más el símbolo ".", más una palabra o número más un símbolo "." cero o una vez, más una palabra o número cero o una vez, más una palabra o número cero o una vez, más una palabra o número cero o una vez. En caso de cumplir el patrón, el resultado se guarda como "*emailAddress*" y el valor de la etiqueta "*emailAddress*" se ingresa a la lista de instancias.

En la **Figura 4-16** se presenta una regla para extraer nombres de organizaciones. El patrón especifica que, si existe una palabra que es un determinante más otra palabra(s) de tipo NPJKINST o si la palabra se encuentra en la lista "Organización", entonces se ingresa la palabra(s) a la lista de instancias y la vincula a la clase organización. La regla permite utilizar cualquiera de las dos opciones; si no la encuentra en las listas predefinidas, busca con el patrón que se define. Otras reglas para extraer instancias se presentan en el **Anexo B**.

```
Phase:Instancias22 /*Nombre de la Fase*/
Input:  Token Lookup SpaceToken /*Tipos de anotaciones de entrada */
Options: control = appelt /* Método que se debe aplicar en caso de superposición de reglas*/


Rule:Instancias22 /*Nombre de la regla */
Priority: 50 /*Prioridad de la regla */
/*Parte izquierda de la regla.*/
(
 ({Token.kind == word}|{Token.kind == number})+ /*Token tipo palabra o número, uno o más veces*/

 ({Token.string == "_"})? /*String "_", cero o una vez */
 ({Token.string == "."})? /*String ".", cero o una vez */
 ({Token.kind == word}|{Token.kind == number}|{Token.string == "_"})* /*Token tipo palabra, o número, o "_", cero o muchas veces */

 {Token.string == "@"} /*String "@"*/
 ({Token.kind == word}|{Token.kind == symbol}|{Token.kind == punctuation}|{Token.kind == number})/*Token tipo palabra, o símbolo, o ".", o número*/
 ({Token.string == "."})? /*String ".", cero o una vez*/
 ({Token.kind == word}|{Token.kind == symbol}|{Token.kind == punctuation}|{Token.kind == number})* /*Token tipo palabra, o símbolo, o ".", o
número, cero o más veces*/
 ({Token.string == "."})? /*String ".", cero o una vez*/
 ({Token.kind == word}|{Token.kind == symbol}|{Token.kind == punctuation}|{Token.kind == number})? /*Token tipo palabra, o símbolo, o ".", o
número, cero o una vez*/
 ({Token.string == "."})? /*String ".", cero o una vez*/
(
{Token.string == "."} /*String "."*/
({Token.kind == word}|{Token.kind == number}) /*String ".", cero o una vez*/
({Token.string == "."})? /*String "."*/
  ({Token.kind == word}|{Token.kind == number})? /*Token tipo palabra o número, cero o una vez*/
  ({Token.string == "."})? /*String ".", cero o una vez*/
  ({Token.kind == word}|{Token.kind == number})? /*Token tipo palabra o número, cero o una vez*/
 )
/*Etiqueta del patrón completo*/
 ):emailAddress
-->
/*Parte derecha de la regla: Si se cumple el patrón de lado izquierdo se guarda la instancia con algunas características*/
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("emailAddress"); /*Se obtiene el conjunto de elementos etiquetados con emailAddress*/
FeatureMap features =Factory.newFeatureMap(); /*Se crean las características*/
for(Annotation ann:as) { /*Ciclo para recorrer todos los elementos encontrados*/
/*Se guarda en la variable instanceName la información del elemento encontrado desde la posición inicial hasta la final*/
String instanceName = doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind")); /*Se le da un tipo al elemento encontrado*/
features.put("rule","Instancias22"); /*Se almacena información de la regla*/
features.put("className","email"); /*El nombre de la clase encontrada*/
features.put("attributeName", "has_name"); /*Nombre del atributo*/
features.put("representationId",ann.getId()); /*Información del ID asignado al elemento en la caracterítica representationId*/
features.put("id",ann.getId()); /*Información del ID asignado al elemento*/
features.put("corefchain", null); /*Información sobre correferencia, no se utiliza*/
features.put("instanceName", instanceName); /*El valor de la palabra(s) tal como se ecnontró en el texto*/
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features); /*Todos los elementos encontrados, se agrupan en la etiqueta Instancias*/
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

Figura 4-15: Regla 22 para etiquetar Instancias de E-mail. **Fuente:** Elaboración Propia

```
Phase:Instancias27 /*Nombre de la Fase*/
Input:  Token Lookup Organization NPJKINST /*Tipos de anotaciones de entrada */
Options: control = appelt /* Método que se debe aplicar en caso de superposición de reglas*/

Rule: Instancias27 /*Nombre de la regla */
Priority: 25 /*Prioridad de la regla */
/*Parte izquierda de la regla.*/
(
({Token.string==~"[Tt]he",Token.category==DT}{NPJKINST.kind=="NPJKINST"})
|
{Organization}
/*Etiqueta del patrón completo*/
):inst
-->
/*Parte derecha de la regla: Si se cumple el patrón de lado izquierdo se guarda la instancia con
algunas características*/
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("inst"); /*Se obtiene el conjunto de elementos
etiquetados con inst*/
FeatureMap features =Factory.newFeatureMap();/*Se crean las características*/
for(Annotation ann:as) { /*Ciclo para recorrer todos los elementos encontrados*/
/*Se guarda en la variable instanceName la información del elemento encontrado desde la posición
inicial hasta la final*/
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind")); /*Se le da un tipo al elemento encontrado*/
features.put("rule","Instancias27"); /*Se almacena información de la regla*/
features.put("className","organization"); /*El nombre de la clase encontrada*/
features.put("attributeName", "has_name"); /*Nombre del atributo*/
features.put("representationId",ann.getId()); /*Información del ID asignado al elemento en la
caracteritica representationId*/
features.put("id",ann.getId()); /*Información del ID asignado al elemento*/
features.put("corefchain", null); /*Información sobre correferencia, no se utiliza*/
features.put("instanceName", instanceName); /*El valor de la palabra(s) tal como se encontró en el
texto*/
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features); /*Todos los elementos
encontrados, se agrupan en la etiqueta Instancias*/
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

Figura 4-16: Regla 27 para etiquetar Instancias de Organización. **Fuente:** Elaboración Propia

4.11.3 Reglas para extracción de Atributos

En esta sección se presenta la extracción de valores de atributos que pertenecen a una instancia de una clase. Uno de los principales aportes de esta Tesis Doctoral se relaciona con la importancia que tienen los adjetivos en la extracción de valores de atributos. Por ejemplo, en la frase "*the red car*", que contiene tres categorías gramaticales (un determinante, un adjetivo y un sustantivo), se debe detectar que "*car*" es una clase y que "*red*" es un valor de atributo que pertenece a esa clase. Para el ejemplo, las entidades ontológicas se deben expresar de la siguiente forma:

- Clase: *car*
- Atributo: *has_color*

- Instancia: *car1*
- Valor de atributo "*color*" en la instancia "*car1"*: *red*
- Expresión ontológica completa: "*car1 has_color red"*

Además, se definen varias categorías mediante una clasificación de adjetivos, de modo que se puedan detectar valores de atributos escondidos en los adjetivos. La clasificación se elaboró basado en una categorización presentada en [69]. Se tomaron los adjetivos más frecuentes en el idioma inglés según la fuente de información y en el caso de algunas categorías se toma ventaja de GATE. Por ejemplo, la categoría *Numeric* tiene una lista predefinida donde se encuentran todos los números ordinales y cardinales, con ello, no se deben escribir todos los números dentro de la implementación de la regla en JAPE, con invocar la lista es suficiente. En los experimentos realizados se encontraron la mayoría de los adjetivos que existían en los textos y sin importar el dominio. En la **Tabla 4-1** se presenta la clasificación de adjetivos desarrollada.

Se decidió crear una propia taxonomía porque permite asignar directamente desde el adjetivo y sustantivo encontrado el nombre del nuevo atributo que se crea en la clase, esta funcionalidad no está implementada en las taxonomías existentes. También para poder nombrar los atributos de las clases que se encuentran en los textos. Por ejemplo, si en un texto se encuentra la frase "six children", el resultado será "children has_number six". De la misma forma se consideró que al tener almacenados los adjetivos más frecuentes en el idioma inglés, con ello se podrían crear mejores nombres de atributos para las clases. El impacto de crear el atributo has_unknown_type es la de informar que aunque se desconoce la categoría del valor del atributo, esta instancia pertenece a la clase o sustantivo que sigue de ese adjetivo. El sistema todavía no tiene la capacidad de ingresar adjetivos a la clasificación, pero será una mejora para una próxima versión.

Tabla 4-1: Clasificación de adjetivos. **Fuente:** Elaboración propia.

TIPO DE ADJETIVO	ADJETIVO
Appearance	*Beautiful, ugly, clean, dirty, complex, safe, dangerous, strong, weak, same, neat, messy, rich, poor, amorphous, scenic*
Authenticity	*Real, actual, authorised*
Brightness	*Dark, bright, shadowy, drab, radiant, shining, pale, dull, glowing, shimmering, luminous, gleaming*

Tabla 4-1: Clasificación de adjetivos. **Fuente:** Elaboración propia.

TIPO DE ADJETIVO	ADJETIVO
Color	*Pink, red, orange, yellowish, dark-green, blue, purple, black, white, gray, brown, tanned, pastel, metallic, silver, colorless, transparent, translucent.*
Condition	*Crazy, sane, sick, healthy, drunk, sober, tired, broken, dead, alive, hungry, asleep, awake, busy, idle, open, closed, married, engaged, separated, divorced, mixed, regulatory, disruptive, terrorist, better, careful, clever, famous, gifted, inexpensive, mushy, powerful, shy, tender, uninterested, expensive, live-music*
Cooking	*Cooked, baked, fried, boiled, peeled, sliced, stewed, steamed, roast, broiled, cut, grated*
Difficulty	*Difficult, hard, easy, simple*
Dimension	*Long, short, little, big, small, large, tall, wide, deep, fourth-quarter*
Distance	*Far, distant, nearby, close, faraway, outlying, remote, far-flung, neighboring, handy*
Distributive	*Each, every, either, neither*
Domain	*Local, general, national, social, international, regional, oceanic, lunar, polar, equatorial, domestic, institutional, academic*
Feelings	*Worried, frustrated, distressed, disappointed, upset, depressed, confused, bored, ashamed, annoyed, afraid, hopeful, thrilled, grateful, pleased, energetic, inspired, enthusiastic, ecstatic, satisfied, content, confident, Amused, angry, bewildered, clumsy, defeated, embarrassed, fierce, grumpy, helpless, itchy, lazy, mysterious, nervous, obnoxious, panicky, repulsive, scary, Agreeable, brave, calm, delightful, eager, faithful, gentle, jolly, kind, lively, obedient, proud, relieved, silly, thankful, witty, lovable*
Justification	*Odd, wrong, right, fair, proper*
Materials	*Iron, Steel, rubber, paper, wooden, plastic, stone, glass, leather, silver, gold, tin, cotton, metal, non-metallic, cloth, concrete, fabric, ceramic, silicon, explosive, sole*
Numeric	***Cardinal:*** *one, two, three.* ***Ordinal:*** *First, second, third.*
Opinion	*Best, worse, worst, wonderful, splendid, mediocre, awful, fantastic, pretty, wasteful, comfortable, uncomfortable, valuable, worthy, worthless, useful, useless, evil, angelic, rare, scarce, disgusting, amazing, surprising, pointless, pertinent, party-on*
Origin	*European, latin, greek, French, American, german, Colombian, Spanish, Italian, Russian, Korean, chinese, Japanese, dutch, argentine, Mexican, australian, Canadian, English, Irish, Floridian, northern.*
Personality	*Honest, courageous, optimistic, intelligent, sincere, ambitious, modest, sensible, friendly, practical, considerate, tolerant, responsible, generous, disciplined, humorous, sympathetic, dishonest, pessimistic, miserly, coward, selfish, impatient, patient, lazy, greedy, resentful, envious, jealous, possessive, conceited, arrogant, fussy, gullible, stubborn, careless, happy, sad, excited, scared, frightened, outgoing, zany, grumpy, cheerful, jolly, carefree, quick-witted, blissful, lonely, elated, autonomous*
Position	*High, low, high-profile*
Primacy	*Only, particular, main, special, major*

Tabla 4-1: Clasificación de adjetivos. **Fuente:** Elaboración propia.

TIPO DE ADJETIVO	ADJETIVO
Purpose	*Folding, swinging, work, racing, cooking, sleeping, dance, rolling, pwalking, knowing, deceiving, binding*
Qualification	*Light, helpful, bloody, human, sorry, sure, able, certain, clear, free, next, true, possible, usual, correct, unusual, specific, legislated, explicit, irregular, regular, subject, standard, unscreened, commercial, strategic, final, consultative, direct, resultant, streamlined, systemic, overt, consistent, radical, basic, likely, extensive, decentralized, unnecessary, efficient, adequate, original, costly, overseas, advisory, legislative, authorized, layered, unlawful, public, dignified, required, projected, degradable, blood-sample, reasonable, alternative, driftless, optimum, respective, random, relative, soil-test, mean, traditional, seedbed, corn-based, split-plot, long-term, subsequent, private, genetic, short-term, continuous, inherent, independent, economic, key, crop-input, favorable, on-farm, mental, predictive, previous, cognitive, effective, professional, maladaptive, self-talk, average, deaf, hoarse, positive, relevant, negative, appropriate, unknown, cheap, false, accurate, complete, athletic, pure, wrinkly, worldly, wise, willful, wild, violent, vicious, utter, unfortunate, triumphant, tremendous, thoughtful, thorough, terrible, suspicious, surly, successful, strict, smart, sheepish, sedate, searching, ripe, rigid, rightful, righteous, restful, reproachful, reluctant, reassuring, ready, ravenous, quintessential, queenly, potential, playful, ,partial, orderly, near, natural, mocking, miserable, meaningful, mad, knightly, kingly, inward, intense, intent, instant, innocent, incredible, immediate, hopeless, holy, gleeful, ghostly, ghastly, garden, furious, frightful, frenetic, frantic, frank, foolish, fervent, ferocious, extreme, excellent, exact, enormous, dreamy, diligent, desperate, dear, curious, cross, continual, common, coaxing, brisk, bold, bashful, anxious, affectionate, absentminded, self-propelled, parental, fiscal.*
Quantity	*Much, any, few, several, most, all, enough, sufficient, abundant, empty, heavy, numerous, double, sparse, substantial, half, no, insufficient, more, plenty, majority, rest, multiple, triple, quantitative, minimum, overall, robust, multilateral, full, single, whole, some, many, extra, countless*
Religion	*Catholic, protestant, anglican, baptist, christian, hindu, Buddhist, muslin, jewish, lutheran*
Shape	*Square, round, rectangular, triangular, oval, conical, spherical, cubical, cylindrical, straight, curved, crooked, flat, steep, hollow, circular, sleek, blobby, rotund, globular, wavy, oblong, elliptical, zigzag, squiggly, winding, serpentine, warped, distorted, broad, skinny, serried*
Similarity	*Other, different, similar, like*

Tabla 4-1: Clasificación de adjetivos. **Fuente:** Elaboración propia.

TIPO DE ADJETIVO	ADJETIVO
Size	*Thin, thick, Big-boned, chubby, flat-boned, giant, gigantic, huge, inmense, jumbo, majestic, mammoth, massive, miniature, petite, puny, scrawny, teeny, tiny, vast, medium, narrow, shallow, various, colossal, fat, immense, teeny-tinytiny*
Smell	*Perfumed, acrid, putrid, burnt, smelly, reeking, noxious, pungent, aromatic, fragrant, scented, musty, sweet-smelling*
Sound	*Loud, silent, vociferous, screaming, shouting, thunderous, blaring, quiet, noisy, talkative, rowdy, deafening, faint, muffled, mute, speechless, whispered, hushed, Cooing, hissing, melodic, purring, raspy, screeching, thundering, voiceless, whispering, inaudible, audible*
Speed	*Quick, fast, rapid, slow, swift, speeding, rushing, bustling, snappy, whirlwind, hasty, prompt, brief*
State	*Liquid, solid, gas, eutectic, associate, civil, unstable, sterile, intermediate, preventive, ongoing, amino, crucial*
Taste	*Sweet, salty, sour, bitter, greasy, fresh, stale, tasty, delicious, tasteless, fatty, rotten, spicy, acidic, savory, delectable, yummy, bland, palatable, luscious, appetizing, watery, juicy, nutritious, tart*
Temperature	*Hot, cold, freezing, icy, frigid, sweltering, wintry, frosty, frozen, nippy, chilly, sizzling, scalding, burning, feverish, fiery, steaming, cool, warm*
Texture	*Rough, wet, slippery, sticky, even, sharp, blunt, tight, loose*
Time	*Old, young, new, modern, ancient, updated, outdated, senior, junior, current, past, future, yearly, monthly, present, further, annual, temporal, contemporary, elderly, ultimate, timely, baby, babyish, teenage, antique, old-fashioned, youthful, mature, adolescent, infantile, bygone, recent, early, late, last, prehistoric*
Touch	*Melted, prickly, uneven, soft, silky, velvety, bumpy, smooth, grainy, coarse, pitted, scaly, polished, glossy, lumpy, wiry, scratchy, glassy, Boiling, breeze, creepy, cuddly, curly, damaged, dusty, flaky, fluffy*
Type	*Shelf-type, port-type, top, artistic, down, federal, dramatic, carry-on, risk, in-flight, on-board, cohesive, intelligence-based, in-depth, industry-wide, aviation-related, rapid-deployment, cost-effective, top-notch, sexual*
Use	*Electrical, political, psychological, physical, historical, financial, medical, technical, logical, internal, external, mechanical, philosophical, environmental, industrial, agronomic, statistical, seasonal, ecological, structural, wrongful, criminal, pharmaceutical, critical, operational, additional, agricultural, empirical, topical, universal, educational, technological, collegial, official, lexical, central*
Value	*Good, nice, great, alright, bad, important, lovely, fine, funny, interesting, okay*
Weather	*Rainy, stormy, sunny, windy, snowy, damp, dry, foggy, overcast, cloudy, mild, atmospheric, wind-lashed*

NOTA: Si el adjetivo no se encuentra en alguna de las categorías, y aprovechando que con las reglas *JAPE se* reconocen todos los adjetivos en un texto, entonces se crea un atributo llamado "*has_unknown_type*". De esta forma, se asegura que se reconocerán todos los adjetivos que preceden los sustantivos, aunque no se les asigne una categoría. Gracias a las categorías planteadas y los adjetivos establecidos, la lista de atributos en la categoría desconocido no es muy amplia y tampoco tienen mucha variación.

Con la clasificación de adjetivos, se procede a especificar las reglas en el lenguaje JAPE. En la **Figura 4-17** se presenta la regla para la categoría "*Authenticity*". En una macro se tienen almacenados los valores de atributos para esta categoría, ellos son: "*real", "actual", "authorised".* Luego en la parte izquierda de la regla se especifica que, si existe un adjetivo antes de un tipo "NPJK" (sustantivo), ese adjetivo es un valor de atributo. En la parte derecha de la regla se define que la clase es el valor de "NPJK", el valor del atributo es el que se encuentra en el texto y el nombre de atributo al que pertenece es "*has_ Authenticity*".

En la **Figura 4-18** se presenta la regla para la categoría "*Position*". Por ejemplo, en la frase "*the high executive*" se detectan tres categorías gramaticales: determinante, adjetivo y sustantivo. Al aplicar la regla, las entidades ontológicas resultantes son:

- Clase: *executive*.
- Atributo: *has_position*
- Instancia: *executive1*
- Valor de Atributo: *high*

```
Phase:Authenticity /*Nombre de la Fase*/
Input:  Token Lookup NPJK /*Tipos de anotaciones de entrada */
Options: control = appelt /* Método que se debe aplicar en caso de superposición de reglas*/

/*MACRO, la cual es llamada muchas veces en la regla*/
Macro:Atrib_authenticity
(
{Token.string==~"[Rr]eal",Token.category==JJ}|{Token.string==~"[Aa]ctual",Token.category==JJ}|{Token
.string==~"[Aa]uthorised",Token.category==JJ}
)

Rule: Authenticity /*Nombre de la regla */
Priority: 35 /*Prioridad de la regla */

/*Parte izquierda de la regla.*/
(
(Atrib_authenticity):sust2 /*VALOR DEL ATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE A LA QUE PERTENECE*/

/*Etiqueta del patrón completo*/
):inst
-->
/*Parte derecha de la regla: Si se cumple el patrón de lado izquierdo se guarda la instancia con
algunas características*/
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2"); /*Se obtiene el conjunto de elementos
etiquetados con sust2*/
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1"); /*Se obtiene el conjunto de elementos
etiquetados con sust1*/

FeatureMap features =Factory.newFeatureMap(); /*Se crean las características*/

/* SE EXTRAE EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase); /*Nombre de la clase*/
}

/*SE EXTRAE EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Authenticity");/*Nombre de la regla*/
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName); /*Valor del Atributo*/
features.put("attributeName", "has_Authenticity"); /*Nombre del atributo*/
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features); /*Todos los elementos encontrados,
se agrupan en la etiqueta Instancias*/

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

Figura 4-17: Regla *Authenticity* para etiquetar valores de atributos. **Fuente:** Elaboración Propia

```
Phase:Test29 /*Nombre de la Fase*/
Input:  Token Lookup NPJK /*Tipos de anotaciones de entrada */
Options: control = appelt /* Método que se debe aplicar en caso de superposición de reglas*/

/*MACRO, la cual es llamada muchas veces en la regla*/
Macro:Atrib_position
(
{Token.string==~"[Hh]igh",Token.category==JJ}|{Token.string==~"[Ll]ow",Token.category==JJ}|{Token.stri
ng==~"[Hh]igh-profile",Token.category==JJ}
)

Rule: Position /*Nombre de la regla */
Priority: 35 /*Prioridad de la regla */
/*Parte izquierda de la regla.*/
(
(Atrib_position):sust2 /*VALORATRIBUTOINSTANCIA*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
/*Etiqueta del patrón completo*/
):inst
-->
/*Parte derecha de la regla: Si se cumple el patrón de lado izquierdo se guarda la instancia con
algunas características*/
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2"); /*Se obtiene el conjunto de elementos
etiquetados con sust2*/
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1"); /*Se obtiene el conjunto de elementos
etiquetados con sust1*/

FeatureMap features =Factory.newFeatureMap(); /*Se crean las características*/

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase); /*Nombre de la clase*/
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Position"); /*Nombre de la regla*/
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName); /*Valor del Atributo*/
features.put("attributeName", "has_position"); /*Nombre del atributo*/
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features); /*Todos los elementos encontrados,
se agrupan en la etiqueta Instancias*/

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

Figura 4-18: Regla *Position* para etiquetar valores de atributos. **Fuente:** Elaboración Propia

Otras reglas para extraer valores de atributos se presentan en el **Anexo C**.

4.11.4 Reglas para extracción de Relaciones

La extracción de relaciones es el proceso mediante el cual se extrae una clase dominio y una clase rango, también se extrae el nombre de la relación entre las dos clases. En la **Figura 4-19** se presenta una regla para extraer una relación. En el patrón se define que, si se encuentra en el texto una palabra(s) tipo "NPJK" que hace referencia a la clase dominio, después se debe encontrar el texto "*caused by*", el cual es el nombre de la relación y, finalmente, se debe encontrar otra(s) palabra(s) tipo "NPJK", que es(son) la(s) clase(s) rango.

La lista de relaciones que reconoce el sistema son: "*caused by"*, "*and other*", "*is a*", "*is located in*", "*derives from*", "*contained in*", "*part whole*", "*preceded by*". Algunos ejemplos de relaciones que se pueden presentar son: *the problem caused by people, the cat is an animal*, *the city located in country, energy derives from sun, approaches part whole relations, month preceded by day*, *pages contained in web*.

Otras reglas para extraer relaciones se presentan en el **Anexo D**.

```
Phase:Test1 /*Nombre de la Fase*/
Input:  Token Lookup NPJK /*Tipos de anotaciones de entrada */
Options: control = appelt

Rule: Relaciones4 /*Nombre de la regla */
Priority: 25 /*Prioridad de la regla */

/*Parte izquierda de la regla*/
(
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE DOMINIO*/
{Token.string=="caused"}
{Token.string=="by"}
({Token.length>1,NPJK.kind=="NPJK"}):sust2 /*CLASE RANGO*/

/*Etiqueta del patrón completo*/
):inst
-->
/*Parte derecha de la regla: Si se cumple el patrón de lado izquierdo se guarda la instancia con
algunas características*/
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");/*Clase Rango*/
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");/*Clase Dominio*/

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE DOMINIO*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA CLASE RANGO Y EL NOMBRE DE LA RELACIÓN*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Relaciones4");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);/*Nombre de la clase rango*/
features.put("relationName", "caused_by"); /*Nombre de la relación*/
}
outputAS.add(as.firstNode(),as.lastNode(),"Relaciones",features); /*Todos los elementos encontrados,
se agrupan en la etiqueta Relaciones*/

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

Figura 4-19: Regla *4* para etiquetar relaciones. **Fuente:** Elaboración Propia

4.12 Proceso *Exporter*

Todos los procesos descritos hasta el momento permiten obtener como resultado una serie de etiquetas con información importante sobre clases, instancias, atributos y relaciones. En el proceso *Exporter* se envía la información sobre etiquetas a un archivo con extensión "*XML*". En la **Figura 4-20** se presenta un extracto de código XML generado con la frase "*the red car*". El código muestra que, al procesar con la regla "Clases15", se extrae el nombre de la clase llamada "*car*". También que se extrae un atributo

"*has_color*", al cual se asigna el valor de atributo "*red*". El archivo XML es necesario para la siguiente y última etapa del método propuesto.

```
<Feature>
  <Name className="java.lang.String">rule</Name>
  <Value className="java.lang.String">Clases15</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">category</Name>
  <Value className="java.lang.String">JJ</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">category</Name>
  <Value className="java.lang.String">NN</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">className</Name>
  <Value className="java.lang.String">car</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">attributeName</Name>
  <Value className="java.lang.String">has_color</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">instanceName</Name>
  <Value className="java.lang.String">red</Value>
</Feature>
```

Figura 4-20: Código XML generado desde *EXPORTER*. **Fuente:** Elaboración Propia

4.13 Proceso *ExtractorJK*

El proceso *ExtractorJK* se implementa en lenguaje Java y se encarga de compactar todos los elementos mediante la creación de una ontología *OWLIM* con sus entidades correspondientes para, posteriormente, generar un archivo con extensión *OWL* y el cual se puede modificar en cualquier editor de ontologías. En la **Figura 4-21** se presenta el algoritmo que describe los pasos del proceso *ExtractorJK*. Es requisito indispensable de esta fase el archivo *XML* que se genera en el proceso *Exporter*.

Algoritmo: ExtractorJK

Data: archivo XML .
Result: $(archivo\ OWL)$

1 Crear una ontología **OWLIM**;
2 Cargar el archivo XML;
3 Procesar el conjunto de anotaciones llamado **Clases**;
4 Procesar el conjunto de anotaciones llamado **Instancias**;
5 Procesar el conjunto de anotaciones llamado **Relaciones**;
6 Definir posibles subclases;
7 Generar archivo con extensión OWL;
8 **return** $(archivo\ OWL)$;

Figura 4-21: Algoritmo *ExtractorJK*. **Fuente:** Elaboración Propia

Según el proceso, primero se crea una ontología *OWLIM,* donde se almacenan las entidades ontológicas conforme a como se procesan. Después, es necesario cargar al sistema el archivo *XML,* el cual es la base de procesamiento en esta fase. Con estos elementos, se empieza a procesar cada una de las entidades ontológicas de acuerdo con las anotaciones que se reciben en el archivo *XML*.

Inicialmente, se procesan las clases. Se recibe un conjunto de anotaciones bajo la etiqueta llamada “Clases”. En la primera tarea se extrae cada elemento del conjunto y se almacena en una estructura tipo lista, la cual facilita el procesamiento. Después, se busca en la lista si existen nombres de clases repetidas y, de ser así, se eliminan y se deja una clase solamente. También, se cambian caracteres especiales como espacios en blanco y saltos de línea por el caracter ”_”. Luego, se cambian los nombres de clases que contengan letras mayúsculas, por letras minúsculas. Para culminar el procesamiento de esta fase, se toma cada elemento de la lista y se crea una clase para cada uno dentro de la ontología.

Posteriormente, se procesan las instancias. Se recibe un conjunto de anotaciones bajo el nombre de “Instancias”. El conjunto contiene información sobre nombre de atributo, valor de atributo y clase a la que pertenece. Para recibir la información se definen tres listas paralelas (véase la **Figura 4-22**), una para los nombres de atributo, otra para los valores de atributos y la última para las clases a la que pertenecen. Después las tres listas se procesan con la función que permite eliminar caracteres especiales y con la función de convertir letras mayúsculas en minúsculas.

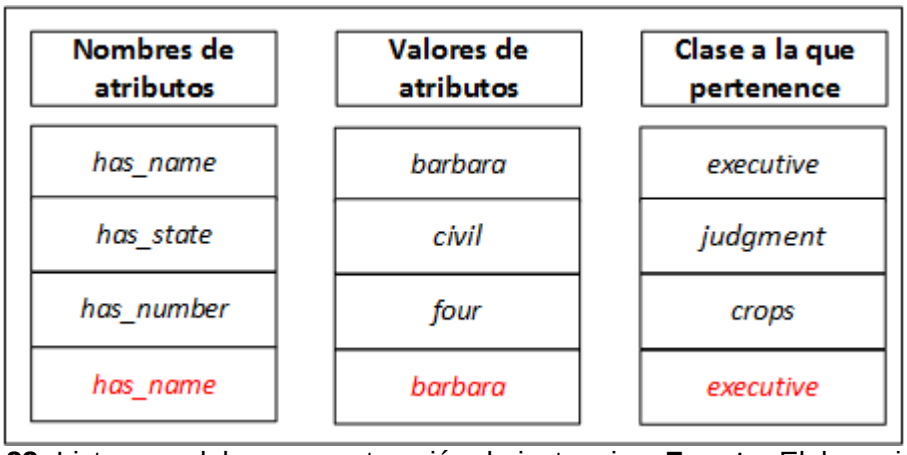


Figura 4-22: Listas paralelas para extracción de instancias. **Fuente:** Elaboración Propia

Seguidamente, una función examina las tres listas de forma paralela con el objetivo de eliminar los valores que se encuentren duplicados (véanse los campos rojos en la **Figura 4-22**) para dejar un solo valor en cada una de las listas. Luego, el sistema verifica que todas las clases de la lista estén creadas como clases en la ontología pues, de lo contrario, estas clases se deben crear. Después, se realiza un proceso iterativo, el cual busca la clase de la lista en la ontología; cuando la encuentra, le asigna a la clase dentro de la ontología el nombre del atributo contenido en la lista. Luego, se crea una instancia genérica y para nombrarla se utiliza el nombre de la clase, más el símbolo "_" y un número. Finalmente, se utiliza el nombre del atributo en esa instancia dentro de la ontología, para asignar el valor del atributo contenido en la lista. Así, mediante el proceso iterativo se realiza la población de la ontología.

Posteriormente, se procesan las relaciones. Se recibe un conjunto de anotaciones con la etiqueta "Relaciones". El conjunto contiene información sobre el nombre de la clase dominio, nombre de la clase rango y nombre de la relación entre las clases. Es necesario crear tres listas paralelas, donde cada una contiene información correspondiente. Las tres listas se procesan con la función para cambiar caracteres especiales como espacios en blanco y saltos de línea por el caracter "_". También, se procesan con la función para convertir caracteres en mayúscula por caracteres en minúscula. Después, se comienza un proceso iterativo, es decir, se realiza para todos los elementos de las listas paralelas al mismo tiempo. Se busca en la ontología el nombre de la clase dominio que se encuentra en la lista y luego se hace lo mismo para la clase rango. Seguidamente, se busca en la posición correspondiente dentro de la lista el nombre de la relación. Con la

identificación de los tres elementos, se procede a crear la relación que permite vincular la clase dominio con la clase rango dentro de la ontología. El proceso se realiza para todos los elementos de las tres listas paralelas.

En este punto, la ontología cuenta con clases, instancias, valores de atributos y relaciones. El proceso siguiente es tratar de identificar las clases que pueden ser subclases. Se realiza un recorrido por la lista de clases, para encontrar qué palabras contienen otras. Por ejemplo, desde de un texto en lenguaje natural, se pueden extraer las clases "*defense*" y "*defense_team*". En este caso "*defense*" es una clase y "*defense_team*" se convierte en una subclase de ella. El sistema, mediante un proceso iterativo, examina esta particularidad en todas las clases existentes y realiza los ajustes pertinentes en la ontología.

Finalmente, después del proceso descrito, por medio del proceso *ExtractorJK,* se genera la ontología con todas las entidades ontológicas que se extraen, lo que permite la creación de un archivo con extensión *OWL* con la información. El archivo se puede leer en cualquier editor de ontologías, como por ejemplo *Protégé*.

5. Validación del Trabajo

Para la validación de esta Tesis Doctoral, se utiliza el método no empírico denominado comparación de características [46,47], que consiste en realizar una comparación de métodos encontrados en la literatura para la población de ontologías para ver cómo cada uno de ellos aborda el mismo problema y comparar resultados. Se utiliza esta técnica porque permite comparar los valores de *precision*, *recall* y *F-measure* de cada uno de los métodos encontrados en la revisión de literatura, con el método que se propone en esta Tesis Doctoral. Este Capítulo contiene cuatro Secciones: en la primera, se realiza un análisis comparativo entre algunos trabajos de la literatura y el método propuesto; en la segunda Sección se realiza una síntesis del método propuesto en diferentes dominios; en la tercera, se realiza un comparativo entre los diferentes métodos encontrados en la literatura y el método propuesto; en la cuarta, se generan tres ontologías desde el lenguaje natural, en la quinta Sección se presentan los productos de nuevo conocimiento, en la sexta se presentan la ventajas y desventajas del método y en la séptima sección las amenazas a la validez. Algunos documentos de prueba se presentan en el **Anexo E**.

5.1 Comparación con otros trabajos

En esta Sección se presenta un análisis comparativo con los trabajos de otros autores y se toman como referencia los criterios de *precision* y *recall*. De esta forma, se pretende evaluar el desempeño del método propuesto.

5.1.1 Trabajo IJntema *et al.* 2012

Los autores [19] aclaran que no es de su interés trabajar ontologías detalladas y extensas, porque sólo les interesa identificar la frecuencia de los eventos de esos dominios. El sistema se alimenta con dos ontologías que crean expertos en los dominios

financiero y político. Para sus experimentos toman diez clases de cada dominio, para evaluar cuántas instancias de relaciones y de clases pueden extraer.

En el dominio financiero utilizan las clases: *CEO*, *product*, *shares*, *competitor*, *profit*, *loss*, *partner*, *subsidiary*, *president* y *revenue*. En el dominio de política utilizan las clases: *election*, *visit*, *sanction*, *join*, *resignation*, *investment*, *riots*, *collaboration*, *provocation* y *help*. Para estas clases, el sistema que se presenta obtiene un valor de *precision* del 80% y *recall* de 70%.

Los autores suministraron algunos de los documentos de prueba que se utilizan para las evaluaciones. Al azar se escoge un documento de prueba por cada dominio, es decir, uno para dominio de finanzas y otro de política. En el documento de finanzas, se logran extraer manualmente 75 instancias y en el documento de política, 38 instancias sin tener en cuenta los valores de atributos que se encuentran en los adjetivos.

Para evaluar el sistema propuesto, se tienen en cuenta los valores de los atributos de instancias que están en los adjetivos, lo que aumenta la exhaustividad del proceso. En el caso del documento de dominio financiero se cuentan manualmente 75 instancias más 30 en los adjetivos, para un total de 105 instancias en el corpus. Con esto y al aplicar una regla de tres simple inversa, los resultados de los autores cambian. El valor de *precision* sigue siendo 80%, el valor de *recall* baja de 70% a 50% y *F-measure* baja de 75% a 61.9%.

En el documento de dominio político, se cuentan manualmente 38 instancias más 18 en los adjetivos, para un total de 56 instancias en el corpus. Para este caso al aplicar una regla de tres simple inversa, los resultados cambian de la siguiente forma: *precision* 80%, *recall* pasa de 70% a 49% y *F-measure* disminuye de 75% a 60%.

Luego, se utilizan los mismos documentos para evaluar el método propuesto en esta Tesis. En el documento de dominio en finanzas, se cuentan un total de 91 instancias correctamente extraídas y 95 instancias extraídas. Es decir, se obtiene un 95.79% de *precision*, 86.67% de *recall* y un *F-measure* de 91%. En el dominio de política se cuentan 50 instancias correctamente extraídas y 54 instancias extraídas, el sistema obtiene *precision* de 92.59%, *recall* de 89.29% y *F-measure* de 90.91%.

5.1.2 Trabajo Ríos 2013

El autor [62] propone un proceso de aprendizaje de ontologías, donde se realizan varias tareas como extracción de conceptos, obtención de relaciones taxonómicas y obtención de axiomas. Realizan pruebas con varios documentos de páginas Web, en el dominio de turismo. Ellos obtienen un valor de *precision* de 56.78% en la extracción de instancias y no presentan resultados de *recall* y *F-measure*.

Se logra obtener un documento de prueba del autor, el cual se encuentra dentro del contenido del trabajo. Se utiliza el método propuesto para evaluar el documento, y se obtiene un valor de *precision* del 95%, un *recall* de 88.37% y *F-measure* de 91.57%. Se debe aclarar que, en el proceso de evaluación, se cuentan los valores de atributos de las instancias los cuales se encuentran en los adjetivos.

5.1.3 Trabajo Faria *et al.* 2013

Los autores [13] definen la tarea de reconocimiento de entidades nombradas, como la identificación de nombres de objetos en el mundo, tales como nombres de personas, organizaciones, lugares, nombres propios y fechas. En ninguna parte del trabajo se hace mención a los valores de atributos de las instancias, los cuales se pueden encontrar en los adjetivos.

El trabajo se valida con la utilización de un corpus en dos dominios, el dominio legal y el dominio de turismo. De acuerdo con el trabajo que presentan los autores, se obtiene desde la web un documento de prueba del dominio legal. El sistema propuesto por los autores obtiene un *precision* de 90%, *recall* de 89.50% y *F-measure* de 89.74%.

En el documento de dominio legal se cuentan manualmente 136 instancias, más 51 de los adjetivos, para un total de 187 instancias en el corpus. En esas condiciones y al aplicar una regla de tres simple inversa, los resultados que presentan los autores en cuanto al valor de *precision* sigue siendo 90%, *recall* cambia de 89.50% a 65.24% y F-measure pasa de 89.74% a 75.78%. Luego, se utiliza el mismo documento para evaluar el método propuesto con un número total de 165 instancias correctamente extraídas y 180 instancias extraídas. El sistema obtiene 91.67% de *precision*, 88.24% de *recall* y *F-measure* de 89.92%.

5.1.4 Collection SMS for a public research CORPUS

Con el fin de realizar otro tipo de evaluación al sistema propuesto, se utiliza un corpus que provee la Universidad Nacional de Singapur (NUS). Este corpus contiene 10117 mensajes de texto o SMS (*Short Message Service*). Para efectos de conteo de las instancias, se toma una muestra de 5%, es decir, 506 SMS. El texto tiene varios errores de escritura y semánticos, pero se quiso examinar el comportamiento del sistema al procesar este tipo de textos. Dado que en los mensajes cortos se utiliza un lenguaje con poca estructura, la extracción no es tarea sencilla, sin embargo, el método obtiene 92.63% de p*recision,* 87,50% de *recall* y 89.99% de F-*measure*.

5.1.5 Collection Twitter messages CORPUS

Las redes sociales tienen un gran impacto actualmente en la vida de las personas. Se considera importante probar el sistema propuesto en la extracción de entidades ontológicas desde estos ambientes. Con este fin, se toma un corpus de mensajes de *Twitter* desde la página de la Universidad de Stanford. El corpus tiene 497 mensajes y, por efectos de conteo manual de instancias, se toma una muestra de 10%, es decir, 49 mensajes. Al igual que sucede con los mensajes SMS tienen errores léxicos y semánticos.

Finalmente, el método identifica instancias con un nivel de *precision* del 95.24%, *recall* del 90.91% y *F-measure* del 93.02%. Se puede concluir que, a pesar de los errores en el texto, el sistema extrae instancias desde un lenguaje con muy poca estructura o que se acerca al lenguaje natural.

5.2 Síntesis de la propuesta en diferentes dominios

Para probar el desempeño del método propuesto en esta Tesis Doctoral, se utilizan doce dominios diferentes para obtener datos más aproximados sobre la extracción de instancias. La Tabla 5-1 presenta los criterios bajo los cuales se realiza la evaluación, y los valores que se obtienen en el proceso. **NICE** representa el número de instancias correctamente extraídas, **NIE** representa el número de instancias extraídas y **NIC** representa el número de instancias en el corpus. Al realizar un promedio entre los dominios, el sistema obtiene un 94% de precision, 89,56% de recall y 91,72% de F-measure.

Tabla 5-1: Síntesis de la propuesta en diferentes Dominios. **Fuente:** Elaboración propia.

Criterios / Dominio	NICE	NIE	NIC	*precision*	*recall*	*F-measure*
Freeze Drying Domain	39	41	42	95,12%	92,86%	93,98%
Showiz Domain	78	80	83	97,50%	93,98%	95,71%
Agriculture Domain	560	630	650	88,89%	86,15%	87,50%
Mental imagery Domain	49	51	54	96,08%	90,74%	93,33%
Air Security Domain	67	72	75	93,06%	89,33%	91,16%
Tuorism Domain	38	40	43	95,00%	88,37%	91,57%
Finance Domain	91	95	105	95,79%	86,67%	91,00%
Politics Domain	50	54	56	92,59%	89,29%	90,91%
Music Domain	68	72	75	94,44%	90,67%	92,52%
Legal Domain	165	180	187	91,67%	88,24%	89,92%
SMS Domain	490	529	560	92,63%	87,50%	89,99%
Twitter Domain	80	84	88	95,24%	90,91%	93,02%
Totales				**94,00%**	**89,56%**	**91,72%**

En cada dominio se evaluó un documento donde se contaron manualmente las instancias posibles y con ello se pudo determinar el nivel de *precision, recall, F-measure*, esto también permitió evaluar que las instancias encontradas por el sistema eran aproximadas a la realidad. En los diferentes dominios se encontraron instancias correspondientes a nombres de personas y de ciudades, correos electrónicos, páginas web, citas de autores, valores de temperaturas ambientales, preguntas, ítems en listas e instancias separadas por conjunciones, entre otras.

5.3 Comparativo entre diferentes métodos para la población de ontologías

A continuación se presenta la **Tabla 5-2**, en la cual se realiza un comparativo entre algunos métodos encontrados en la literatura sobre población de ontologías y el método que se presenta en esta Tesis.

Tabla 5-2: Comparativo entre métodos para población de ontologías. **Fuente:** Elaboración propia.

Criterios / Autor	***Dominio Gral/Específico***	***precision***	***recall***	***F-measure***	***¿Construye Ontología?***
Contreras 2004 [15]	General (Relaciones Geopolíticas)	--	--	--	Si
Pasca 2004 [18]	General (5 Categorías)	88,0%	--	--	No
Geleijnse and Korst 2005 [16]	General (Cine)	78,0%	93,8%	85,17%	No
De Boer *et al.* 2007 [54]	Específico (Fútbol y patrimonio cultural)	--	--	--	No
Yoon *et al.* 2007 [55]	Específico (Dispositivos electrónicos)	95,2%	--	--	No
Talukdar *et al.* 2008 [11]	General (5 Clases)	77,4%	--	--	No
Manine *et al. 2008* [56]	Específico (Biomédico)	89,6%	89,3%	89,44%	No
Ruiz-Martínez *et al.* 2008 [57]	Específico (Turismo)	93,2%	94,9%	94,04%	No
Danger y Berlanga 2009 [58]	General (Arqueología)	90,0%	90,0%	90,0%	No
Faria y Girardi 2011 [49]	Específico (Derecho de Familia)	95,0%	75,0%	83,82%	No
Schlaf y Remus 2012 [59]	Específico (4 Categorías)	84,9%	45,0%	58.82%	No
IJntema *et al.* 2012 [19]	Específico (Finanzas y Política)	80,0%	70,0%	74.66%	No
Ruiz-Martinez *et al.* 2012 [60]	Específico (Biomédico)	79,6%	69,0%	73.92%	No
Faria *et al.* 2012 [17]	General (Turismo y Legal)	81,9%	82,0%	81,94%	No
De Araujo *et al.* 2013 [12]	Específico (Legal)	98,0%	91,5%	94,63%	No
Sadoun *et al. 2013* [61]	Específico (Espacios inteligentes)	95,0%	63,0%	75,75%	No
Ríos 2013 [62]	Específico (Turismo)	56,8%	--	--	Si
Faria *et al. 2013* [13]	General (Turismo y Legal)	87,3%	86,5%	86,89%	No
Lima *et al.* 2014 [14]	General (Nombres de ciudades, enfermedades, aves, pescados y mamiferos)	94,0%	59,0%	72,49%	No

Tabla 5-2: Comparativo entre métodos para población de ontologías. **Fuente:** Elaboración propia.

Criterios / Autor	***Dominio Gral/Específico***	***precision***	***recall***	***F-measure***	***¿Construye Ontología?***
Nederstig *et al.* 2014 [20]	Específico (E-Commerce)	96,0%	89,0%	92,36%	No
Colace *et al.* 2014 [63]	General	--	--	--	Si
Santos y Girardi 2014 [64]	Específico (Derecho de Familia)	--	--	--	Si
Kordjamshidi y Moens 2015 [10]	General	--	--	--	No
Blandón and Zapata 2016	General	94%	89,56%	91,72%	Si

La tabla muestra que algunos métodos alcanzan niveles más altos de *precision*, *recall* y *F-measure* en comparación con el método propuesto. Sin embargo, algunos de esos métodos utilizan un dominio específico, no generan ontologías automáticamente con todos sus elementos desde el lenguaje natural y otros no tienen en cuenta los valores de atributos que se encuentran escondidos en los adjetivos que anteceden sustantivos, lo cual podría disminuir los niveles de *recall* y *F-measure* de esos métodos.

5.4 Generación de ontologías

En esta Sección, se presenta la generación de ontologías directamente desde el lenguaje natural. Para realizar la primera evaluación, se toma el siguiente fragmento de texto: "***A self-propelled vehicle is a motor vehicle or road vehicle that does not operate on rails***" [70]. En la **Figura 5-1** se presenta la ontología que genera el método propuesto.

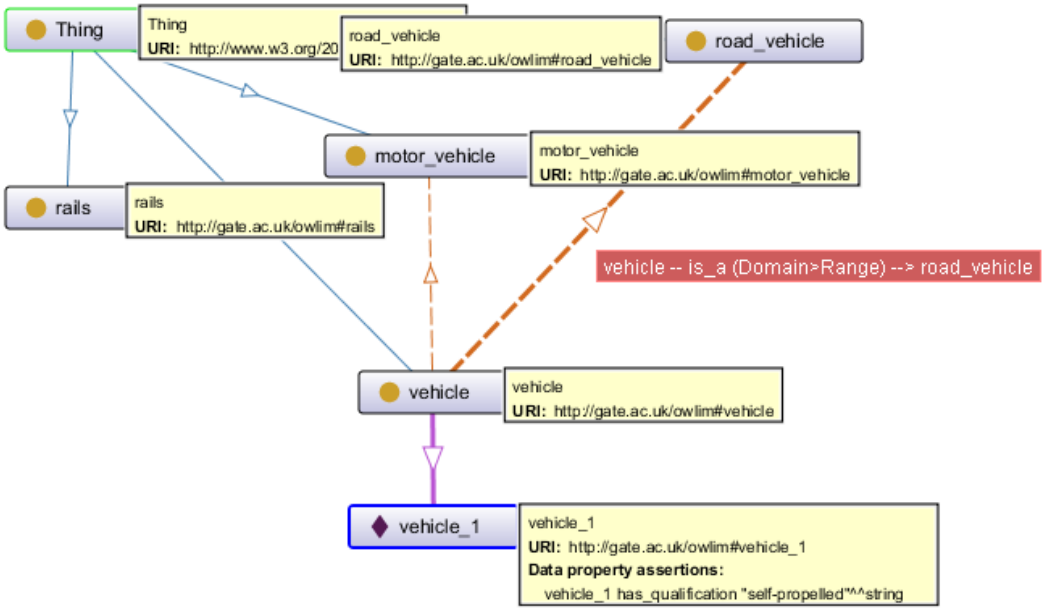


Figura 5-1: Ontología generada desde el lenguaje natural. **Fuente:** Elaboración Propia

Se extraen cuatro clases, "*vehicle*", "*motor_vehicle", "road_vehicle"* y "*rails*". En la clase "*vehicle*" se crea un atributo de clase llamado "*has_qualification*" de tipo *String*. Después, se crea una instancia genérica llamada "*vehicle_1",* y en su campo "*has_qualification*" se asigna el valor de atributo "*self-propelled*". También, se crean dos relaciones: "*vehicle is_a motor_vehicle*" y "*vehicle is_a road_vehicle*".

Luego para realizar una segunda evaluación, se toma el siguiente fragmento de texto: "***The marriage produced six children: Micah, Matthew, Amber, Joseph, Amanda, and Jennifer***" [13]. En la **Figura 5-2** se presenta la ontología que genera el método propuesto. Se extraen 2 clases, "*marriage*" y "*children*". En la clase "children" se crean 7 instancias genéricas llamadas "*children1*", "*children3*", "*children12*", "*children11*", "*children14*", "*children13*", "*children15*". En la clase *"children"* se crean dos atributos llamados "*has_quantity*" y "*has_name*". En la instancia "*children_3*" y en el atributo "*has_quantity*" se asigna un valor de "seis". En la instancia "*children_1*" y en el atributo "*has_name*" se asigna un valor de "jennifer". En la instancia "*children_12*" y en el atributo "*has_name*" se asigna un valor de "mathew". En la instancia "*children_11*" y en el atributo "*has_name*" se asigna un valor de "micah". En la instancia "*children_14*" y en el atributo "*has_name*" se asigna un valor de "joseph". En la instancia "*children_13*" y en el atributo "*has_name*" se asigna un valor de "amber" y finalmente en la instancia "*children_15*" y en el atributo "*has_name*" se asigna un valor de "amanda".

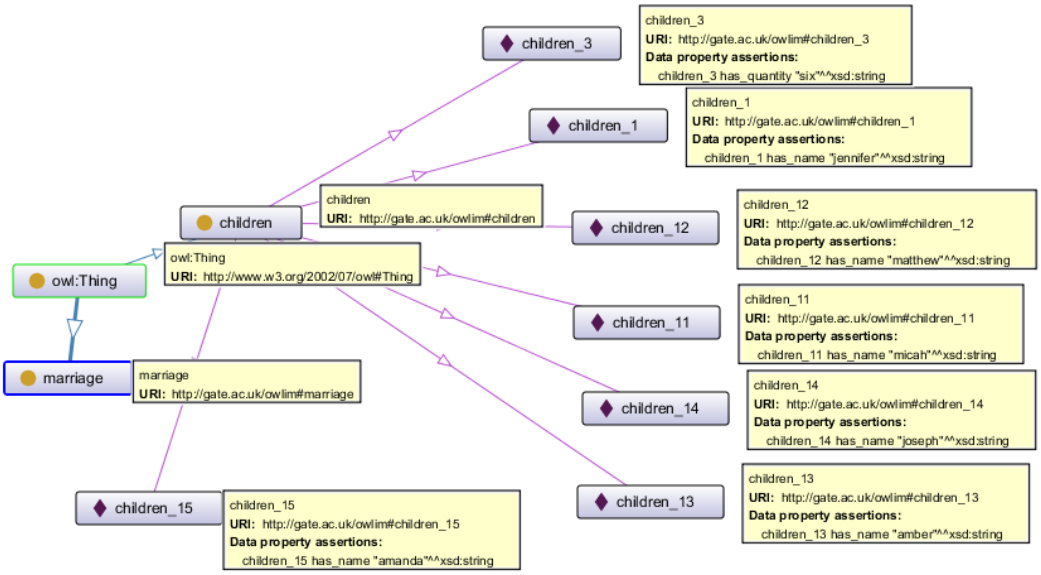


Figura 5-2: Segunda Ontología generada desde el lenguaje natural. **Fuente:** Elaboración Propia

Después se realiza una tercera evaluación, para ello se toma el siguiente fragmento de texto: "***The conference will be attended by CEOs like Steve Ballmer and Steve Jobs***" [19]. En la **Figura 5-3** se presenta la ontología que genera el método propuesto.

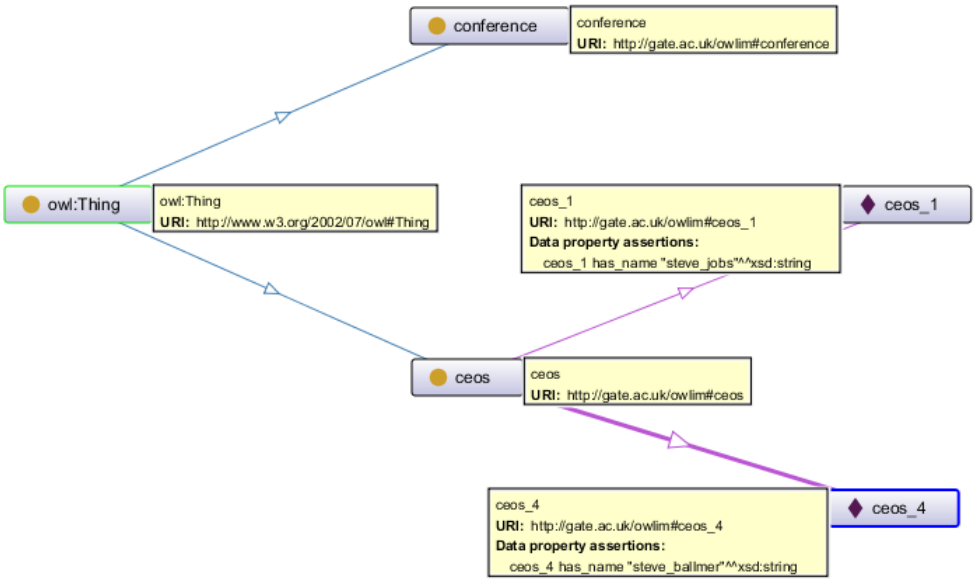


Figura 5-3: Tercera ontología generada desde el lenguaje natural. **Fuente:** Elaboración Propia

Se extraen 2 clases, "*conference*" y "*ceos*". En la clase "ceos" se crean 2 instancias genéricas llamadas "*ceos_1*", "*ceos_4*". En la clase *"ceos"* se crea un atributo llamado "*has_name*". En la instancia "*ceos_1*" y en el atributo "*has_name*" se asigna un valor de "steve_jobs". En la instancia "*ceos_4*" y en el atributo "*has_name*" se asigna un valor de "*steve_ballmer*".

5.5 Productos de nuevo conocimiento

Desde la Tesis Doctoral que se presenta, se generaron distintos productos de nuevo conocimiento. El primero tiene que ver con la posibilidad de estructurar una asignatura a nivel de posgrado sobre la creación, población y enriquecimiento de ontologías desde lenguaje natural, mediante la utilización de patrones JAPE en la arquitectura *GATE* (véase la **Tabla 5-3**). El segundo, es la publicación de artículos académicos; cabe mencionar que por ahora se presenta uno (véase la **Tabla 5-4**), pero cabe aclarar que existe otro artículo en fase de construcción. El tercer producto tiene que ver con el método computacional completo y funcional en GATE, que se presenta en el cuerpo de la Tesis Doctoral y en los anexos.

Tabla 5-3: Propuesta de asignatura a nivel Posgrado. **Fuente:** Elaboración propia.

Asignatura:	Construcción Automática de Ontologías
Fase de Formación:	Posgrado
Créditos:	4
Justificación:	El auge de Internet en los últimos años permite la generación de gran cantidad de información, lo cual permite que las personas puedan realizar consultas desde diferentes lugares y temas. La información que se consulta en la web actual se incluye en documentos y las búsquedas se realizan con palabras clave. Para tratar de conectar esta información y realizar algún tipo de inferencia surgen las ontologías como una evolución de las bases de datos convencionales. Las ontologías permiten crear taxonomías con capacidad de inferencia, es decir, se pretende con ellas dar semántica a la información para darle más sentido, lo cual constituye su aspecto más importante. La creación de estructuras ontológicas de forma manual es un proceso costoso, por lo cual se requieren métodos que permitan la creación de ontologías de forma (semi) automática desde el lenguaje natural.
Objetivos:	**General** Desarrollar conocimiento específico sobre la creación, población y enriquecimiento de ontologías directamente desde el lenguaje natural y utilizando la herramienta GATE-JAPE combinada con el lenguaje programación Java.
	Específicos ▪ Identificar los principales conceptos sobre ontologías y procesamiento de lenguaje natural.

Tabla 5-3: Propuesta de asignatura a nivel Posgrado. **Fuente:** Elaboración propia.

	▪ Reconocer el funcionamiento de la herramienta GATE-JAPE. ▪ Crear métodos básicos para el aprendizaje, población o enriquecimiento de ontologías. ▪ Conectar las aplicaciones GATE-JAPE con el lenguaje Java.
Metodología:	Se plantea un espacio de clase con la participación, la discusión, la comprensión, la compartición de conocimientos y la familiarización de términos con los estudiantes. El punto de partida es el diagnóstico del conocimiento de los estudiantes; luego, se presentan algunas explicaciones magistrales del docente, exposiciones de temas y trabajos de clase; finalmente se entra a la etapa de aclaración de conceptos donde se retroalimentan los temas vistos con trabajos de participación grupal. Se complementa esta metodología con la realización de foros temáticos en el área y la realización de juegos basados en experiencias. La evaluación se realiza con un trabajo práctico por entregas y dos foros temáticos.
***Contenido general*:**	▪ Conceptualización teórica sobre ontologías. ▪ Principales conceptos sobre procesamiento de lenguaje natural. ▪ Arquitectura GATE-JAPE. ▪ Aprendizaje de ontologías. ▪ Población de ontologías. ▪ Enriquecimiento de ontologías.

Tabla 5-4: Producto de conocimiento número 1. **Fuente:** Elaboración propia.

Título:	GATE-BASED PATTERNS FOR AUTOMATICALLY EXTRACTING ATTRIBUTE VALUES
Tipo:	Artículo Científico
Autores:	Juan Carlos Blandón Andrade y Carlos Mario Zapata Jaramillo
Revista:	Revista Ingeniería Investigación y Tecnología. Universidad Nacional Autónoma de México
Indexación:	A1 de Publindex
Estado:	En revisión
Año:	2016
***Abstract*:**	Automated ontology population is necessary for enriching ontologies and facilitating the search for information in the Web. Among the main entities of an ontological model, we can find classes, subclasses, attributes—datatype properties—, relationships—object properties—, and instances. Class instances are important to the scientific community, since a lot of work is devoted to automatically populating ontologies by using statistical methods, information extraction, and natural language processing, among others. The problem is focused on identifying and extracting attribute values for instances. Commonly, such values have a predefined type like numeric, string, boolean, etc. The difficulty arises when you want to know which instance belongs to such values. In this paper we propose an approach based on Natural Language Processing (NLP) and Information Extraction (IE) technologies for extracting attribute values. We use syntactic patterns implemented on the GATE (General Architecture for Text Engineering) tool. The results are independent of the application domain and they exhibit promising values of recall, precision, and F-measure.
***Keywords*:**	**Automatic ontology population, natural language processing, information extraction, GATE-JAPE patterns.**

5.6 Ventajas y desventajas del método propuesto

El método automático para población de ontologías desarrollado en esta Tesis Doctoral, cuenta con las siguientes ventajas:

- El método logra poblar ontologías de forma automática sin importar el dominio de aplicación, recibiendo como entrada un archivo en formato PDF o TXT y obtiene buenos niveles de *precision, recall* y *F-measure*.
- Cuenta con la capacidad de extraer clases e instancias simples o compuestas, valores de atributos, relaciones entre clases y otras entidades ontológicas que son etiquetadas.
- Las entidades ontológicas etiquetadas se envían con éxito a un archivo XML, el cual será el insumo principal para el *plugin* que construye la ontología**.**
- El *plugin* ExtractorJK desarrollado en lenguaje Java y teniendo como base el archivo XML, construye la ontología y la almacena en un archivo con extensión OWL.
- El sistema realiza el procesamiento de documentos de extensión de una página en 1,578 segundos.

Se pueden citar las siguientes desventajas:

- La Interfaz gráfica poco amigable al usuario. Este aspecto se pretende mejorar para versiones futuras con la creación de una aplicación Web, en la cual las personas puedan ingresar sus documentos y extraer sus elementos ontológicos.
- El sistema tiene problemas para procesar 200 páginas o más, en algunas ocasiones entra a un bucle infinito. La forma de mitigar este problema es la creación de una función que termine el proceso en estas situaciones, así se evita consumir los recursos de la máquina.
- En algunos casos el sistema no reconoce los caracteres especiales en los textos, esto hace que el etiquetado de las palabras quede con algunas inconsistencias. Este aspecto se abordará en futuras versiones del método debido a que se pueden presentar documentos en malas condiciones.

5.7 Amenazas a la validez del método

La validez de una investigación científica se puede relacionar a la veracidad de sus resultados y a consecuencia de ello las conclusiones de esa investigación, tendrán más fuerza. Según [66] el objeto de estudio puede experimentar varios mecanismos que pueden entorpecer o invalidar la inferencia causal, esto también es llamado amenazas a la validez de los resultados.

Aunque existen muchas amenazas, durante el desarrollo de esta Tesis Doctoral se mitigó de alguna manera el hecho de la selección de los textos a evaluar y desde los cuales se realizaría la población de ontologías. Para que el autor de esta Tesis no seleccionara textos predefinidos y sencillos, el Director del trabajo enviaba textos aleatorios y que a su juicio eran complicados para realizar la extracción. La dinámica consistía en planear una reunión semanal donde el autor de la Tesis realizaba ajustes al método y luego el Director enviaba un texto totalmente desconocido para el sistema, luego se procesaba y se medía el porcentaje de los niveles de *precision, recall* y *F-measure* del sistema en ese texto.

También pensando en que la selección del texto por parte del Director tuviese algún sesgo, se decidió tomar textos de prueba de otros autores que también presentaran trabajos de población de ontologías. El resultado de procesamiento del sistema no varió mucho y los resultados seguían siendo fiables. Finalmente, se tomaron corpus de twitter y mensajes de texto (SMS), que son un lenguaje natural, el sistema aunque tuvo algunos problemas con caracteres especiales, logró extraer los elementos ontológicos.

6. Conclusiones y Trabajo Futuro

La inteligencia artificial es una línea de investigación en constante evolución, muchos investigadores enfocan sus trabajos en la creación de entidades inteligentes para automatizar tareas intelectuales y, así, tratar de emular la forma en que el ser humano las realiza. Las ontologías son entidades que tratan de expresar formalmente el conocimiento modelando un dominio, para después inferir nuevo conocimiento. Existe una disciplina llamada ingeniería ontológica, que se encarga de construir herramientas para agilizar el proceso de creación de ontologías desde el lenguaje natural y que tiene tres tareas fundamentales (la construcción, población y enriquecimiento automático de ontologías), los cuales son campos de interés para la comunidad académica. El método propuesto se enfoca en la población de ontologías desde el lenguaje natural y con un buen nivel de eficacia. Adicionalmente a ello, se implementa la posibilidad de aprendizaje de ontologías, es decir, extraer conceptos/clases y relaciones automáticamente desde el lenguaje natural. Al obtener las ontologías completas, se puede compartir conocimiento, reutilizar y organizar el conocimiento, mejorar búsquedas de información (en la web) y se abre una grande opción para programar agentes inteligentes. En este Capítulo se presentan las conclusiones con los aportes relevantes de la investigación y el trabajo futuro con posibles líneas de trabajo sobre la tarea de población de ontologías.

6.1 Conclusiones

La importancia de la población de ontologías, radica en que las ontologías se deben actualizar constantemente pues, de lo contrario, sería información estática. También, existen muchas ontologías con dominios especializados que se pueden enriquecer con muchas instancias. Finalmente, la población de ontologías es un aporte importante, por ejemplo, para mejorar las búsquedas en Internet. La población de ontologías de forma manual desde lenguaje natural, es una tarea complicada, costosa y que consume mucho tiempo. Según la revisión de literatura, es necesario crear métodos semi-automáticos o

automáticos que permitan realizar esta tarea con mayor eficiencia. También, se evidencia la necesidad de crear métodos que permitan realizar el proceso sin importar dominio. Adicionalmente, la extracción de las entidades ontológicas se debe realizar desde cualquier fuente de información y con buenos niveles de *precision*, *recall* y *F-measure*.

En esta Tesis Doctoral se propone un método computacional que, de acuerdo con la problemática detectada, permite avanzar en el área de creación de ontologías mediante los siguientes aportes:

- Revisión de literatura con los principales conceptos relacionados con la población de ontologías y los enfoques recientes más utilizados para la construcción de métodos (semi) automáticos.

- Hallazgos relacionados con los adjetivos que anteceden sustantivos, los cuales contienen valores de atributos. Este detalle no se menciona en la mayoría de trabajos que se utilizan como base teórica de esta Tesis. Este aspecto hace que los resultados que presentan diferentes autores bajen en sus niveles de exhaustividad, por lo tanto, pueden disminuir su *recall* de 15 a 20% y *F-measure* de 10 a 15% aproximadamente.

- Diseño e implementación de un sistema de reglas genéricas, mediante patrones sintácticos en lenguaje JAPE, que se utilizan para la extracción de entidades ontológicas como clases, atributos, instancias y relaciones, directamente desde el lenguaje natural con independencia del dominio.

- Definición de una clasificación de adjetivos para el idioma inglés y una implementación de dicha clasificación en lenguaje JAPE, con lo cual se logran identificar muchos valores de atributos, que se encuentran en textos de lenguaje natural.

- Desarrollo de un *Plugin* en Lenguaje Java y compatible con la arquitectura GATE, el cual permite que se puedan extraer las anotaciones definidas mediante el sistema de reglas, para luego exportar todas las entidades ontológicas a una ontología general y posteriormente crear el archivo con extensión *OWL.* En otras

palabras, permite conectar todos los procesos descritos del método, para que se pueda compilar y ejecutar bajo un ambiente de programación. Esta característica transversal no se pudo detectar en los trabajos analizados, que se limitaban a uno u otro de los procesos.

- Diseño e implementación de un método computacional automático, que recibe como entrada un archivo en lenguaje natural de extensión PDF o TXT. Con base en este archivo se realiza la construcción y población de una ontología y luego se genera como salida un archivo con extensión OWL, el cual contiene entidades ontológicas, tales como clases simples y compuestas, subclases, atributos, instancias, relaciones y todo con independencia del dominio. El archivo que se genera se puede modificar en cualquier editor de ontologías.

- Extracción de instancias correspondientes a nombres de personas y de ciudades, correos electrónicos, páginas web, citas de autores, valores de temperaturas ambientales, preguntas, ítems en listas e instancias separadas por conjunciones, entre otras.

- Aplicación del método no empírico denominado comparación de características para la validación de la propuesta, en la cual se utilizan doce dominios diferentes y se obtienen resultados globales de 94% de *precision,* 89,56% de *recall* y 91.72% de *F-measure.*

Estos aportes ayudan para el avance en la construcción de herramientas automáticas, que puedan llevar a cabo los procesos de construcción y población de ontologías. La arquitectura GATE demuestra que tiene un buen desempeño en el procesamiento del lenguaje natural y que al estudiar muy bien su funcionamiento se pueden obtener buenos resultados.

Entre las mayores dificultades encontradas durante la elaboración de esta Tesis Doctoral, se encuentran; (i) el diseño de reglas genéricas para poder procesar textos en diferentes dominios, debido a los aspectos complejos del lenguaje natural; (ii) la existencia de poca documentación sobre la arquitectura GATE y el lenguaje JAPE, pues la comunidad de

desarrolladores existente es pequeña y existen pocos recursos (*blogs* y otros) para apoyar la solución de problemas.

6.2 Trabajo Futuro

De acuerdo al desarrollo de esta Tesis Doctoral y con base a todo el aprendizaje a lo largo de la misma, se proponen algunas líneas para trabajo futuro en la automatización de los procesos de aprendizaje y población de ontologías. Algunas propuestas son:

- Crear un método computacional de extracción de instancias para el idioma español, utilizando las bases fundamentales que se establecen en esta Tesis Doctoral.

- Extender las reglas sintácticas y genéricas en lenguaje JAPE propuestas en esta Tesis Doctoral, para lograr que haya un mayor enriquecimiento de instancias de clases y relaciones.

- Diseñar e implementar un método computacional en la herramienta NLTK de Python que, según los autores, brinda simplicidad, consistencia, extensibilidad y modularidad, para poder medir resultados y comparar la eficiencia respecto del método planteado en esta Tesis Doctoral.

- Proponer un método computacional que utilice técnicas de extracción de información, procesamiento de lenguaje natural con reglas genéricas y aprendizaje de máquina, para medir el desempeño y evaluar qué tanto afecta la carga computacional de las tres técnicas juntas.

A. Anexo: Reglas JAPE para extracción de clases

Reglas Generales (NPJK)

```
Phase:NPJK
Input: Token
Options: control = brill

Rule:NPJK1
Priority: 25
(
({Token.length>1,Token.category==NN})
):nounPhrase
-->
:nounPhrase.NPJK = {kind="NPJK",rule=NPJK1}
```

```
Phase:NPJK
Input: Token
Options: control = brill

Rule:NPJK2
Priority: 35
(
({Token.length>1,Token.category==NN}{Token.length>1
,Token.category==NN})
):nounPhrase
-->
:nounPhrase.NPJK={kind="NPJK",rule=NPJK2}
```

```
Phase:NPJK
Input: Token
Options: control = brill

Rule:NPJK4
Priority: 45
(
({Token.length>1,Token.category==NN}{Token.length>1
,Token.category==NN}{Token.length>1,Token.category=
=NN}{Token.length>1,Token.category==NN})
):nounPhrase
-->
:nounPhrase.NPJK={kind="NPJK",rule=NPJK4}
```

```
Phase:NPJK
Input: Token
Options: control = brill

Rule:NPJK5
Priority: 45
(
({Token.length>1,Token.category==NNS})
):nounPhrase
-->
:nounPhrase.NPJK = {kind="NPJK",rule=NPJK5}
```

```
Phase:NPJK
Input: Token
Options: control = brill

Rule:NPJK7
Priority: 45
(
({Token.category==NN}{Token.category==NNS})
):nounPhrase
-->
:nounPhrase.NPJK={kind="NPJK",rule=NPJK7}
```

```
Phase:NPJK
Input: Token
Options: control = brill

Rule:NPJK8
Priority: 45
(
({Token.category==RB}{Token.category==VBD}{Token.category==NNS})
):nounPhrase
-->
:nounPhrase.NPJK={kind="NPJK",rule=NPJK8}
```

Reglas Específicas

```
Phase:Test1
Input:  Token Lookup
Options: control = appelt

Rule: Clases1
Priority: 25
(
{Token.length>1,Token.kind==word,Lookup.minorType==country}
{Token.category==NN,Token.length==1,Token.kind==punctuation,Token.position==endpunct} /* */
{Token.category==VBZ,Token.length==1,Token.string=="s"}
({Token.length>1,Token.kind==word,Token.category==NN}):sust1

):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust1");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toStr
ing();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Clases1");
features.put("className",ann.getFeatures().get("class"));
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(),as.lastNode(),"Clases",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test1
Input:  Token Lookup
Options: control = appelt

Rule: Clases2
Priority: 25
(
({Token.category == PRP}|{Lookup.class == Person})
{Token.category =~ VB}
{Token.category == DT}
({Token.length>1,Token.category == NN}):sust1
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust1");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toStr
ing();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Clases2");
features.put("className",ann.getFeatures().get("class"));
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(),as.lastNode(),"Clases",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test1
Input:  Token Lookup
Options: control = appelt

Rule: Clases4
Priority: 25
(
{Token.kind==word,Token.category==DT}
{Token.kind==word,Token.category==VBN}
({Token.length>1,Token.kind==word,Token.category==NN}):sust1
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust1");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toS
tring();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Clases4");
features.put("className",ann.getFeatures().get("class"));
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(),as.lastNode(),"Clases",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test1
Input:  Token Lookup
Options: control = appelt

Rule: Clases5
Priority: 25
(
{Token.kind==word,Token.category==CC}
{Token.kind==word,Token.category==NN}
{Token.kind==word,Token.category==NN}
({Token.length>1,Token.kind==word,Token.category==NN}):sust1
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust1");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toS
tring();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Clases5");
features.put("className",ann.getFeatures().get("class"));
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(),as.lastNode(),"Clases",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test1
Input:  Token Lookup
Options: control = appelt

Rule: Clases6
Priority: 25
(
{Token.kind==word,Token.category==DT}
({Token.length>1,Token.kind==word,Token.category==NN}):sust1
{Token.kind==word,Token.category==IN}
{Token.kind==word,Token.category==VBG}
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust1");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toS
tring();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Clases6");
features.put("className",ann.getFeatures().get("class"));
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(),as.lastNode(),"Clases",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test1
Input:  Token Lookup
Options: control = appelt

Rule: Clases7
Priority: 25
(
{Token.kind==word,Token.category==DT}
{Token.kind==word,Token.category==VBG}
{Token.kind==word,Token.category==NN}
{Token.kind==word,Token.category==IN}
({Token.length>1,Token.kind==word,Token.category==NN}):sust1

):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust1");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toS
tring();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Clases7");
features.put("className",ann.getFeatures().get("class"));
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(),as.lastNode(),"Clases",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test1
Input:  Token Lookup
Options: control = appelt

Rule: Clases8
Priority: 25
(
{Token.kind==word,Token.category==PRP}
{Token.kind==word,Token.category==VBZ}
({Token.length>1,Token.kind==word,Token.category==NN}):sust1
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust1");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toS
tring();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Clases8");
features.put("className",ann.getFeatures().get("class"));
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(),as.lastNode(),"Clases",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test1
Input:  Token Lookup
Options: control = appelt

Rule: Clases9
Priority: 25
(
{Token.length>1,Token.kind==word,Token.category==IN}
{Token.length>1,Token.kind==word,Token.category==JJ}
({Token.length>1,Token.kind==word,Token.category==NNS}):sust1

):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust1");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toS
tring();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Clases9");
features.put("className",ann.getFeatures().get("class"));
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(),as.lastNode(),"Clases",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test1
Input:  Token Lookup
Options: control = appelt

Rule: Clases10
Priority: 25
(
{Token.length>1,Token.kind==word,Token.category==NNP}
{Token.length>1,Token.kind==word,Token.category==NN}
({Token.length>1,Token.kind==word,Token.category==NNS}):sust1

):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust1");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toS
tring();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Clases10");
features.put("className",ann.getFeatures().get("class"));
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(),as.lastNode(),"Clases",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test1
Input: Token Lookup NPJK NPJKINST
Options: control = appelt

Rule: Clases11
Priority: 25
(
{Token.category==WP}
({Token.category==NN,Token.length>1}):sust2

):inst
-->
:sust2.Clases={rule= "Clases11"}
```

```
Phase:Test1
Input: Token Lookup NPJK
Options: control = appelt

Rule: Clases12
Priority: 25
(
({Token.length>1,NPJK.kind=="NPJK"}):sust1
{Token.kind==word,Token.category==CC}
({Token.length>1,NPJK.kind=="NPJK"}):sust2
):inst
-->
:sust1.Clases={rule= "Clases12"},
:sust2.Clases={rule= "Clases12"}
```

```
Phase:Test1
Input: Lookup Token NPJK
Options: control = appelt

Rule: Clases13
Priority: 25
(
{Token.category==IN,Token.length>1}
({Token.length>1,NPJK.kind=="NPJK"}):sust1

):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust1");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toS
tring();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Clases13");
features.put("className",ann.getFeatures().get("class"));
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(), as.lastNode(),"Clases",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test1
Input:  Token Lookup NPJK
Options: control = appelt

Rule: Clases15
Priority: 25
(
{Token.category==JJ}
({Token.length>1,NPJK.kind=="NPJK"}):sust1
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust1");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Clases15");
features.put("className",ann.getFeatures().get("class"));
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(),as.lastNode(),"Clases",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test1
Input:  Token Lookup NPJK
Options: control = appelt

Rule: Clases16
Priority: 25
(
{Token.category==CD}
({Token.category==NNS}):sust1
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust1");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Clases16");
features.put("className",ann.getFeatures().get("class"));
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(),as.lastNode(),"Clases",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test1
Input: Token Lookup NPJK
Options: control = appelt

Rule: Clases17
Priority: 35
(
({Token.length>1,NPJK.kind=="NPJK"}):sust1
{Token.kind==word,Token.category==CC}
({Token.length>1,NPJK.kind=="NPJK"}):sust2
{Token.kind==word,Token.category==TO}
({Token.length>1,NPJK.kind=="NPJK"}):sust3
):inst
-->
:sust1.Clases={rule= "Clases17"},
:sust2.Clases={rule= "Clases17"},
:sust3.Clases={rule= "Clases17"}
```

B. Anexo: Reglas JAPE para extracción de instancias

Reglas Generales (NPJKINST)

```
Phase:NPJKINST
Input: Token
Options: control = brill

Rule:NPJKINST1
Priority: 20
(
({Token.category==NNP})
):nounPhrase
-->
:nounPhrase.NPJKINST =
{kind="NPJKINST",rule=NPJKINST1}
```

```
Phase:NPJKINST
Input: Token
Options: control = brill

Rule:NPJKINST2
Priority: 35
(
({Token.category==NNP}{Token.category==NNP})
):nounPhrase
-->
:nounPhrase.NPJKINST =
{kind="NPJKINST",rule=NPJKINST2}
```

```
Phase:NPJKINST
Input: Token
Options: control = brill

Rule:NPJKINST3
Priority: 40
(
({Token.category==NNP}{Token.category==NNP}
{Token.category==NNP})
):nounPhrase
-->
:nounPhrase.NPJKINST =
{kind="NPJKINST",rule=NPJKINST3}
```

```
Phase:NPJKINST
Input: Token
Options: control = brill

Rule:NPJKINST4
Priority: 45
(
({Token.category==NNP}{Token.category==NNP}
{Token.category==NNP}{Token.category==NNP})
):nounPhrase
-->
:nounPhrase.NPJKINST =
{kind="NPJKINST",rule=NPJKINST4}
```

```
Phase:NPJKINST
Input: Token
Options: control = brill

Rule:NPJKINST6
Priority: 45
(
({Token.category==NNP,Token.length==1}{Token.category=="."}{Token
.category==NNP,Token.length==1}{Token.category=="."}{Token.catego
ry==NNP})
):nounPhrase
-->
:nounPhrase.NPJKINST = {kind="NPJKINST",rule=NPJKINST6}
```

Reglas Específicas

```
Phase:Instancias20
Input:  Token Lookup Date
Options: control = appelt

Rule: Instancias20
Priority: 25
(
{Date}
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("inst");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias20");
features.put("className","date");
features.put("attributeName", "has_name");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias21
Input:  Token Lookup
Options: control = appelt

Macro:SECONDPART
(
{Token.kind ==word}
{Token.string == "."}
{Token.kind ==word}
{Token.string == "/"}
({Token.kind ==word}|{Token.kind ==number})
)


Rule: Instancias21
Priority: 25
(
   (({Token.string == "http"} | {Token.string == "ftp"})
    {Token.string == ":"}
    {Token.string == "/"}
    {Token.string == "/"}
     (SECONDPART)
   )
        |
  ({Token.string == "www"}
   {Token.string == "."}
  ({Token.kind == word}|
  {Token.kind == number})
   {Token.string == "."}
   ({Token.kind == word} | ({Token.kind == word}{Token.string == "."}{Token.kind == word}))
  )
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("inst");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias21");
features.put("className","website");
features.put("attributeName", "has_name");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias23
Input:  Token Lookup Location
Options: control = appelt

Rule: Instancias23
Priority: 25
(
({Location})
|
({Lookup.majorType==facility}|{Lookup.majorType==facility_key}|{Lookup.majorType==facility_ke
y_extra})
):inst

-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("inst");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toSt
ring();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias23");
features.put("className","Location");
features.put("attributeName", "has_name");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias24
Input: Money
Options: control = appelt

Rule:Instancias24
  (
{Money}
):money
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("money");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).t
oString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias24");
features.put("className","money");
features.put("attributeName", "has_name");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias25
Input:  Token Lookup Person NPJK NPJKINST
Options: control = appelt

Rule: Instancias25
Priority: 25
(
({Person,!Token.category==VB}|{Lookup.majorType==person_full,!Token.category==VB})
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("inst");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias25");
features.put("className","person");
features.put("attributeName", "has_name");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias25a
Input:  Token Lookup Person Location NPJK NPJKINST
Options: control = appelt

Rule: Instancias25a
Priority: 60
(
({Token.category==NNP}{Token.category==NNP,Token.length==1}
{Token.category==".",Token.kind==punctuation,Token.length==1,Token.string=="."}):sust2
{Token.string=="is"}
({Token.length>1,NPJK.kind=="NPJK"}):sust1
):inst
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias25a");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias25b
Input:  Token Lookup Person NPJK NPJKINST
Options: control = appelt

Rule: Instancias25b
Priority: 60
(
({Token.string=="by"})
({Token.length>1,NPJK.kind=="NPJK"}):sust1
({Token.category==IN})
({NPJKINST.kind=="NPJKINST"}):sust2
({Token.category==CC})
({NPJKINST.kind=="NPJKINST"}):sust3
):inst
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase = doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName = doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias25b");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}

try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust3");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase = doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName = doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias25b");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}

}
```

```
Phase:Instancias25c
Input:  Token Lookup Person NPJK NPJKINST Organization
Options: control = appelt

Rule: Instancias25c
Priority: 60
(
({NPJKINST.kind=="NPJKINST"}):sust2
{Token.category==VBZ}
({Organization}|{NPJKINST.kind=="NPJKINST"})
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias25c");
features.put("className","employed");
features.put("attributeName", "has_name");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias25d
Input:  Token Lookup Person NPJK NPJKINST
Options: control = appelt

Rule: Instancias25d
Priority: 60
(
{Token.category==DT}
({NPJKINST.kind=="NPJKINST"}):sust2
({Token.length>1,NPJK.kind=="NPJK"}):sust1
):inst
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias25d");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias25e
Input:  Token Lookup Person NPJK NPJKINST
Options: control = appelt

Rule: Instancias25e
Priority: 60
(
({NPJKINST.kind=="NPJKINST"}):sust2
{Token.string=="is"}
{Token.string=="a"}
({Token.length>1,NPJK.kind=="NPJK"}):sust1
):inst
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias25e");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias25f
Input:  Token Lookup Person NPJK NPJKINST
Options: control = appelt

Rule: Instancias25f
Priority: 60
(
({Token.length>1,NPJK.kind=="NPJK"}):sust1
{Token.string=="such"}
{Token.string=="as"}
({NPJKINST.kind=="NPJKINST"}):sust2

):inst
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias25f");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias25g
Input:  Token Lookup Person NPJK NPJKINST
Options: control = appelt

Rule: Instancias25g
Priority: 60
(
{Token.string=="such"}
({Token.length>1,NPJK.kind=="NPJK"}):sust1
{Token.string=="as"}
({NPJKINST.kind=="NPJKINST"}):sust2
):inst
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias25g");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias25h
Input:  Token Lookup Person NPJK NPJKINST
Options: control = appelt

Rule: Instancias25h
Priority: 60
(
({Token.length>1,NPJK.kind=="NPJK"}):sust1
({Token.string=="especially"}|{Token.string=="including"})
({NPJKINST.kind=="NPJKINST"}):sust2
):inst
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias25h");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias25i
Input:  Token Lookup Person NPJK NPJKINST
Options: control = appelt

Rule: Instancias25i
Priority: 60
(
({NPJKINST.kind=="NPJKINST"}):sust2
({Token.string=="and"}|{Token.string=="or"})
{Token.string=="other"}
({Token.length>1,NPJK.kind=="NPJK"}):sust1
):inst
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias25i");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias25j
Input:  Token Lookup Person NPJK NPJKINST
Options: control = appelt

Rule: Instancias25j
Priority: 60
(
{Token.category==IN}
({NPJKINST.kind=="NPJKINST"}):sust2
({Token.length>1,NPJK.kind=="NPJK"}):sust1
):inst
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias25j");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias25k
Input:  Token Lookup Person NPJK NPJKINST
Options: control = appelt

Rule: Instancias25k
Priority: 60
(
({NPJKINST.kind=="NPJKINST"}):sust2
{Token.category==CC}
{Token.category==DT}
({Token.length>1,NPJK.kind=="NPJK"}):sust1
):inst
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias25k");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias251
Input:  Token Lookup Person NPJK NPJKINST
Options: control = appelt

Rule: Instancias251
Priority: 60
(
(({Token.category==CD}{Token.category==NNP})|({Token.category==CD}{Token.kind==punctuation}{Token.ca
tegory==CD}{Token.category==NNP})):sust2
{Token.string=="of"}
{Token.category==NNP}
({Token.category==NN}):sust1
):inst
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias251");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_quantity");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias25m
Input:  Token Lookup NPJK NPJKINST
Options: control = appelt

Rule: Instancias25m
Priority: 60
(
({Token.length>1,NPJK.kind=="NPJK"}):sust1
({Token.category==":",Token.kind==punctuation,Token.length==1,Token.string==":"})
({NPJKINST.kind=="NPJKINST"}):sust2
({Token.category==",",Token.kind==punctuation,Token.length==1,Token.string==","})
({NPJKINST.kind=="NPJKINST"}):sust3
({Token.category==",",Token.kind==punctuation,Token.length==1,Token.string==","})
({NPJKINST.kind=="NPJKINST"}):sust4
({Token.category==",",Token.kind==punctuation,Token.length==1,Token.string==","})
({NPJKINST.kind=="NPJKINST"}):sust5
({Token.category==",",Token.kind==punctuation,Token.length==1,Token.string==","})
({NPJKINST.kind=="NPJKINST"}):sust6
({Token.category==",",Token.kind==punctuation,Token.length==1,Token.string==","})
{Token.category==CC}
({NPJKINST.kind=="NPJKINST"}):sust7
):inst
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias25m");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
```

```
Phase:Instancias25n
Input:  Token Lookup
Options: control = appelt

Rule: Instancias25n
Priority: 25
(
({Token.category==NNP}{Token.category==NNP,Token.length==1}{Token.category==".",Token.kind==punctuat
ion,Token.length==1,Token.string=="."})
|
({Token.category==NNP,Token.length==1}{Token.category==".",Token.kind==punctuation,Token.length==1,T
oken.string=="."}({Token.category==NNP,Token.length==1}
|{Token.category==NN,Token.length==1}){Token.category==".",Token.kind==punctuation,Token.length==1,T
oken.string=="."})
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("inst");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias25n");
features.put("className","persona");
features.put("attributeName", "has_name");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias25o
Input:  Token Lookup
Options: control = appelt

Rule: Instancias25o
Priority: 25
(
{Token.string=="Judge"}
(({Token.category==NNP})|
({Token.category==NNP}{Token.category==NNP})|
({Token.category==NNP}{Token.category==NNP}{Token.category==NNP})|
({Token.category==NNP}{Token.category==NNP}{Token.category==NNP}{Token.category==NNP})):sust1
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust1");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias25o");
features.put("className","judge");
features.put("attributeName", "has_name");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias25p
Input:  Token Lookup Location Organization NPJKINST
Options: control = appelt

Rule: Instancias25p
Priority: 25
(
{Token.string==~"[Bb]y"}
({NPJKINST.kind=="NPJKINST",!Organization}):sust1
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust1");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias25p");
features.put("className","person");
features.put("attributeName", "has_name");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias28
Input:  Token Lookup
Options: control = appelt

Rule: Instancias28
Priority: 25
(
{Lookup.majorType==jobtitle}
):inst

-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("inst");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias28");
features.put("className","job");
features.put("attributeName", "has_name");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias29
Input:  Token Lookup
Options: control = appelt

Rule: Instancias29
Priority: 25
(
{Lookup.majorType==sport}
):inst

-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("inst");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias29");
features.put("className","sport");
features.put("attributeName", "has_name");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias30
Input:  Token Lookup
Options: control = appelt

Rule: Instancias30
Priority: 25
(
({Token.category==NNP}):sust1
{Token.string=="and"}
({Token.category==NNP}):sust2
{Token.category=="(",Token.kind==punctuation,Token.length==1}
{Token.category==CD}
{Token.category==")",Token.kind==punctuation,Token.length==1}
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust1");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias30");
features.put("className","authors");
features.put("attributeName", "has_name");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
```

```
Phase:Instancias31
Input:  Token Lookup NPJK NPJKINST
Options: control = appelt

Rule: Instancias31
Priority: 25
(
({Token.length>1,NPJK.kind=="NPJK"}):sust1
({NPJKINST.kind=="NPJKINST"}):sust2

):inst
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias31");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias32

Input:  Token Lookup NPJK NPJKINST
Options: control = appelt

Rule: Instancias32
Priority: 25
(
{Token.kind==word,Token.string=="of"}
({Token.length>1,NPJK.kind=="NPJK"}):sust1
({NPJKINST.kind=="NPJKINST"}):sust2
):inst
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias32");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias33
Input:  Token Lookup NPJK NPJKINST
Options: control = appelt

Rule: Instancias33
Priority: 25
(
{Token.kind==word,Token.string=="of"}
({Token.length>1,NPJK.kind=="NPJK"}):sust1
{NPJKINST.kind=="NPJKINST"}
{Token.kind==word,Token.category==CC}
({NPJKINST.kind=="NPJKINST"}):sust2
):inst
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias33");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias34
Input:  Token Lookup NPJK NPJKINST
Options: control = appelt

Rule: Instancias34
Priority: 25
(
{Token.kind==word,Token.string=="the"}
({NPJKINST.kind=="NPJKINST"}):sust2
{Token.kind==word,Token.category==CC}
{NPJKINST.kind=="NPJKINST"}
({Token.kind==word,Token.category==NNS}):sust1
):inst
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");


FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias34");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias35
Input:  Token Lookup NPJK NPJKINST
Options: control = appelt

Rule: Instancias35
Priority: 25
(
{Token.kind==word,Token.string=="the"}
{NPJKINST.kind=="NPJKINST"}
{Token.kind==word,Token.category==CC}
({NPJKINST.kind=="NPJKINST"}):sust2
({Token.kind==word,Token.category==NNS}):sust1
):inst
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");


FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias35");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias36
Input:  Token Lookup NPJK NPJKINST
Options: control = appelt

Rule: Instancias36
Priority: 25
(
({NPJK.kind=="NPJK"}):sust1
{Token.string==",",Token.kind==punctuation}
({NPJKINST.kind=="NPJKINST",Token.length>1}):sust2
):inst
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias36");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Nacion
Input:  Token Lookup
Options: control = appelt

Rule: Instancias38
Priority: 25
(
(
{Token.category == CD,Token.kind==number}
({Token.kind ==word ,Token.string=="degrees"}|{Token.kind ==word ,Token.string=="degree"})
)
|
(
{Token.category == CD,Token.kind==number}
({Token.kind ==word ,Token.string ==~"[Cc]"}|{Token.kind ==word ,Token.string ==~"[Ff]"})
)
|
(
{Token.category ==":",Token.kind==punctuation,Token.length==1,Token.string=="-"}
{Token.category == CD,Token.kind==number}
({Token.kind ==word ,Token.string ==~"[Cc]"}|{Token.kind ==word ,Token.string ==~"[Ff]"})
)
|
(
{Token.category ==":",Token.kind==punctuation,Token.length==1,Token.string=="-"}
{Token.category == CD,Token.kind==number}
{Token.category ==NN,Token.length==1}
({Token.kind ==word ,Token.string ==~"[Cc]"}|{Token.kind ==word ,Token.string ==~"[Ff]"})
)
|
(
{Token.category == CD,Token.kind==number}
{Token.category ==NN,Token.length==1}
({Token.kind ==word ,Token.string ==~"[Cc]"}|{Token.kind ==word ,Token.string ==~"[Ff]"})
)
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("inst");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias38");
features.put("className","temperature");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias39
Input:  Token Lookup
Options: control = appelt

Rule: Instancias39
Priority: 25
(
{Lookup.majorType==elem_names}
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("inst");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias39");
features.put("className","chemical_elements");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias40
Input:  Token Lookup Person
Options: control = appelt

Rule: Instancias40
Priority: 25
(
({Token.string == "University"}
{Token.string == "of"}
{Lookup.minorType == city})
 |
(
{Lookup.minorType == city}
{Token.string == "University"}
)):inst

-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("inst");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias40");
features.put("className","university");
features.put("attributeName", "has_name");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias41
Input:  Token Lookup
Options: control = appelt

Macro: PAISES
(
{Token.string==~"[Ss]cotland"}
)

Rule: Instancias41
Priority: 25
(
(
({Token.string==~"[Bb]ank"}{Token.string=="of"}({Lookup.minorType == country} | (PAISES)))
|
({Lookup.minorType == country}{Token.string==~"[Bb]ank"})
|
({Lookup.minorType == country}{Token.kind==word}{Token.string==~"[Bb]ank"})
)
|
({Lookup.majorType==banks})
):inst

-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("inst");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias41");
features.put("className","bank");
features.put("attributeName", "has_name");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias42
Input:  Token Lookup
Options: control = appelt

Macro:Preposic
(
{Token.category == PRP}
)

Macro:Auxiliares
(
{Token.string==~"[Dd]id"}|{Token.string==~"[Dd]o"}|{Token.string==~"[Dd]oes"}|{Token.string==~"[Ww]ill"}|{Token.string==~"[Hh]ad"}|
{Token.string==~"[Hh]ave"}|{Token.string==~"[Hh]as"}
)

Macro:WH
(
{Token.string==~"[Ww]hat"}|{Token.string==~"[Ww]here"}|{Token.string==~"[Ww]ho"}|{Token.string==~"[Ww]hom"}|{Token.string==~"[Hh]ow
"}|{Token.string==~"[Ww]hy"}
|{Token.string==~"[Ww]hen"}|{Token.string==~"[Ww]hose"}|{Token.string==~"[Ww]hich"}
)

Macro:TOBE
(
{Token.string==~"[Ii]s"}|{Token.string==~"[Aa]re"}|{Token.string==~"[Ii]am"}
)

Macro:PRON
(
{Token.string==~"[Ii]"}|{Token.string==~"[Yy]ou"}|{Token.string==~"[Hh]e"}|{Token.string==~"[Ss]he"}|{Token.string==~"[Ii]t"}|{Toke
n.string==~"[Ww]e"}|{Token.string==~"[Tt]hey"}
)

Macro:VERBO
(
{Token.category == VB} | {Token.category == VBD} |{Token.category == VBG} | {Token.category == VBN} |{Token.category == VBP}
|{Token.category == VBZ}
)

Macro:Sustantiv
({Token.category == NN} | {Token.category == NNS} |{Token.category == NNPS} | {Token.category == NNP}
)

Macro:Adjetiv
(
({Token.category == "JJ"}) |({Token.category == "JJR"})|({Token.category == "JJS"}) |({Token.category == "JJSS"})
)

Macro:Poder
(
({Token.string==~"[Cc]an"}) |({Token.string==~"[Mm]ust"})
)

Rule: Instancias42
Priority: 25
(
//TIPO1
((WH)(TOBE)(PRON){Token.string=="?"})
|
//TIPO2
((WH)(TOBE)((Sustantiv)|{Lookup.class == Person}){Token.string=="?"})
|
//TIPO3
((WH)(Auxiliares)({Lookup.class == Person}|(PRON))(VERBO){Token.string=="?"})
|
//TIPO4
((Auxiliares)((PRON)|{Lookup.class == Person})(VERBO)(Sustantiv){Token.string=="?"})
|
//TIPO5
((Auxiliares)((PRON)|{Lookup.class == Person})(VERBO){Token.string=="?"})
|
//TIPO6
((Auxiliares)((PRON)|{Lookup.class == Person})(Sustantiv){Token.string=="?"})
|
//TIPO7
((WH)(Auxiliares)({Lookup.class == Person}|(PRON))(VERBO)((Sustantiv)|(Adjetiv)){Token.string=="?"})
|
//TIPO8
((Poder)({Lookup.class == Person}|(PRON))(VERBO){Token.string=="?"})
|
//TIPO9
((TOBE)(Adjetiv)(Sustantiv){Token.string=="?"})
|
//TIPO10
({Token.kind==word}{Token.kind==word}{Token.kind==word}{Token.kind==word}{Token.string=="?"})
|
//TIPO11
((WH)(TOBE){Token.category == "DT"}{Token.category == "NN"}{Token.category == "VBN"}{Token.string=="?"})
|
//TIPO12
({Token.category == "WRB"}{Token.category == "VBP"}{Token.category == "JJ"}{Token.category == "NNS"}{Token.category ==
"JJ"}{Token.string=="?"})
|
//Tipo13
({Token.category == "WRB"}{Token.category == "VBZ"}{Token.category == "DT"}{Token.category == "NN"}{Token.category ==
"RB"}{Token.category == "JJ"}{Token.string=="?"})
|
//Tipo14
({Token.category == "WRB"}{Token.category == "VBZ"}{Token.category == "DT"}{Token.category == "NNP"}{Token.category ==
"NN"}{Token.string=="?"})
|
//Tipo15
({Token.category == "VBZ"}{Token.category == "PRP"}{Token.category == "JJ"}{Token.category == "TO"}{Token.category ==
"VB"}{Token.category =="NNS"}{Token.category == "CC"}{Token.category == "NNS"}{Token.category == "IN"}{Token.category ==
"DT"}{Token.category == "NN"}{Token.string=="?"})
|
//Tipo16
({Token.category == "WRB"}{Token.category == "VBZ"}{Token.category == "NN"}{Token.category == "CC"}{Token.category ==
"NN"}{Token.category == "NN"}{Token.string=="?"})
|
//Tipo17
({Token.category == "WP"}{Token.category == "VBZ"}{Token.category == "WRB"}{Token.category == "NN"}{Token.category ==
"CC"}{Token.category == "NN"}{Token.category == "NNS"}{Token.category == "VBP"}{Token.string=="?"})
|
//Tipo18
({Token.category == "WP"}{Token.category == "VBZ"}{Token.category == "DT"}{Token.category == "NNP"}{Token.string=="?"})
|
//Tipo19
({Token.category == "WRB"}{Token.category == "VBP"}{Token.category == "NNS"}{Token.category == "JJ"}{Token.string=="?"})
|
//Tipo20
({Token.category == "IN"}{Token.category == "NN"}{Token.category == "VBG"}{Token.string=="?"})

):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("inst");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName = doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias42");
features.put("className","question");
features.put("attributeName", "has_description");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias44
Input:  Token Lookup
Options: control = appelt

Rule: Instancias44
Priority: 25
(
{Lookup.majorType==color}
):inst

-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("inst");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias44");
features.put("className","color");
features.put("attributeName", "has_name");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias45
Input:  Token Lookup NPJK NPJKINST
Options: control = appelt

Rule: Instancias45
Priority: 25
(
({NPJKINST.kind=="NPJKINST"}):sust2
{Token.category==VBZ}
{Token.category==DT}
{Token.category==JJ}
({NPJK.kind=="NPJK"}):sust1
):inst
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias45");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias46
Input:  Lookup Token
Options: control = appelt

Rule: Instancias46
Priority: 75
(
{Lookup.majorType==river_name}
{Token.string=="river"}
):inst

-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("inst");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias46");
features.put("className","river");
features.put("attributeName", "has_name");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias47
Input:  Lookup Token
Options: control = appelt

Rule: Instancias47
Priority: 35
(
{Lookup.majorType==planet_name}
):inst

-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("inst");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias47");
features.put("className","planet");
features.put("attributeName", "has_name");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias48
Input:  Lookup Token
Options: control = appelt

Rule: Instancias48
Priority: 35
(
{Lookup.majorType==time}
{!Token.string=="'"}
):inst

-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("inst");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias48");
features.put("className","time");
features.put("attributeName", "has_name");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias49
Input:  Lookup Token
Options: control = appelt

Rule: Instancias49
Priority: 35
(
{Lookup.majorType==device}
):inst

-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("inst");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias49");
features.put("className","device");
features.put("attributeName", "has_name");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias50
Input:  Lookup Token
Options: control = appelt

Rule: Instancias50
Priority: 35
(
{Lookup.majorType==technolog}
):inst

-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("inst");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias50");
features.put("className","Technology");
features.put("attributeName", "has_name");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias51
Input:  Token Lookup NPJK NPJKINST
Options: control = appelt

Macro:DECIMAL(
({Token.category==CD})
|
({Token.category==CD}({Token.category==".",Token.length==1}|{Token.category==",",Token.length==1}){T
oken.category==CD})
|
({Token.category==CD}({Token.category==".",Token.length==1}|{Token.category==",",Token.length==1}){T
oken.category==CD}{Token.category==".",Token.length==1}{Token.category==CD})
)

Rule: Instancias51
Priority: 25
(
(DECIMAL):sust2
({NPJK.kind=="NPJK"}):sust1
):inst
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias51");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_quantity");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias52
Input:  Lookup Token NPJKINST
Options: control = appelt

Rule: Instancias52
Priority: 40
(
{Token.category==DT}
({NPJKINST.kind=="NPJKINST"}):sust1
{Token.string=="of"}
{Token.string=="the"}
{NPJKINST.kind=="NPJKINST"}
):inst

-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust1");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias52");
features.put("className","Organization");
features.put("attributeName", "has_name");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias54
Input:  Token Lookup NPJK NPJKINST
Options: control = appelt

Rule: Instancias54
Priority: 25
(
({NPJKINST.kind=="NPJKINST",Token.length>1}):sust2
({NPJK.kind=="NPJK"}):sust1
):inst
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias54");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias55
Input:  Token Lookup Location
Options: control = appelt

Rule: Instancias55
Priority: 25
(
({Token.string == "Airport"}{Token.string == "of"}{Lookup.minorType == city})
|
({Lookup.minorType == city}{Token.string == "Airport"})
):inst

-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("inst");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as) {
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias55");
features.put("className","airport");
features.put("attributeName", "has_name");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias57
Input:  Token Lookup NPJK NPJKINST
Options: control = appelt

Rule: Instancias57
Priority: 25
(
({NPJK.kind=="NPJK"}):sust1
{Token.category==IN}
({NPJKINST.kind=="NPJKINST"}):sust2
{Token.category==",",Token.kind==punctuation,Token.length==1,Token.string==","}
({NPJKINST.kind=="NPJKINST"}):sust3
{Token.category==CC}
({NPJKINST.kind=="NPJKINST"}):sust4
)
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias57");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
```

```
Phase:Instancias59
Input:  Token Lookup NPJK NPJKINST
Options: control = appelt

Rule: Instancias59
Priority: 25
(
({NPJK.kind=="NPJK"}):sust1
{Token.category==",",Token.kind==punctuation,Token.length==1,Token.string==","}
({Token.category==NNP,Token.length==1}{Token.category==CD}{Token.category==NNP,Token.length==1})
{Token.category==CC}({Token.category==NNP,Token.length==1}{Token.category==CD}{Token.category==NNP,T
oken.length==1}):sust2
)
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias59");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}

}
```

```
Phase:Instancias61
Input:  Token Lookup NPJK NPJKINST
Options: control = appelt

Rule: Instancias61
Priority: 25
(
(({Token.category==NNP,Token.length==1}{Token.category==CD}({Token.category==NNP,Token.length==1}|{T
oken.category==DT,Token.length==1})):sust2
{Token.category==CC}
({Token.category==NNP,Token.length==1}{Token.category==CD}{Token.category==NNP,Token.length==1}))
({NPJK.kind=="NPJK"}):sust1
)
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias61");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias62
Input:  Token Lookup NPJK NPJKINST
Options: control = appelt

Macro:DECIMAL(
({Token.category==CD})
|
({Token.category==CD}({Token.category==".",Token.length==1}|{Token.category==",",Token.length==1}){T
oken.category==CD})
|
({Token.category==CD}({Token.category==".",Token.length==1}|{Token.category==",",Token.length==1}){T
oken.category==CD}{Token.category==".",Token.length==1}{Token.category==CD})
)


Rule: Instancias62
Priority: 25
(
{Token.category==IN}
({NPJK.kind=="NPJK"}):sust1
(DECIMAL):sust2
)
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias62");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias63
Input:  Token Lookup NPJK NPJKINST
Options: control = appelt

Macro:DECIMAL(
({Token.category==CD})
|
({Token.category==CD}({Token.category==".",Token.length==1}|{Token.category==",",Token.length==1}){T
oken.category==CD})
|
({Token.category==CD}({Token.category==".",Token.length==1}|{Token.category==",",Token.length==1}){T
oken.category==CD}{Token.category==".",Token.length==1}{Token.category==CD})
)

Rule: Instancias63
Priority: 25
(
{Token.category==IN}
({NPJK.kind=="NPJK"}):sust1(DECIMAL)
{Token.category==CC}(DECIMAL):sust2
)
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias63");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias64
Input:  Lookup Token
Options: control = appelt

Rule: Instancias64
Priority: 35
(
({Token.string=="Hotel"}{Token.category==NNP}{Token.category==NNP}{Token.category==NNP})
|
({Token.string=="Hotel"}{Token.category==NNP}{Token.category==NNP})
|
({Token.string=="Hotel"}{Token.category==NNP})
|
({Token.category==NNP}{Token.string=="Hotels"}{Token.category==NNP})
|
({Token.category==NNP}{Token.string=="Hotels"})
|
({Token.category==NNP}{Token.string=="Hotel"})
|
({Token.category==NNP}{Token.string=="Hotel"}{Token.category==NNP})
|
({Lookup.majorType==name_hotel})
):inst

-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("inst");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias64");
features.put("className","hotel");
features.put("attributeName", "has_name");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias65
Input:  Token Lookup NPJK NPJKINST
Options: control = appelt

Macro:DECIMAL(
({Token.category==CD})
|
({Token.category==CD}({Token.category==".",Token.length==1}|{Token.category==",",Token.length==1}){T
oken.category==CD})
|
({Token.category==CD}({Token.category==".",Token.length==1}|{Token.category==",",Token.length==1}){T
oken.category==CD}{Token.category==".",Token.length==1}{Token.category==CD})
)

Rule: Instancias65
Priority: 25
(
({NPJK.kind=="NPJK"}):sust1
{Token.category==IN}
({Token.category=="$",Token.kind==symbol,Token.length==1,Token.string=="$"})
(DECIMAL):sust2
)
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias65");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_name");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias66
Input:  Token Lookup NPJK NPJKINST
Options: control = appelt

Rule: Instancias66
Priority: 25
(
({Token.category==JJ}):sust2
{Token.category==CC}
{Token.category==JJ}
({NPJK.kind=="NPJK"}):sust1
)
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias66");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_qualification");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias69
Input:  Token Lookup NPJK NPJKINST
Options: control = appelt

Macro:DECIMAL(
({Token.category==CD})
|
({Token.category==CD}({Token.category==".",Token.length==1}|{Token.category==",",Token.length==1}){T
oken.category==CD})
|
({Token.category==CD}({Token.category==".",Token.length==1}|{Token.category==",",Token.length==1}){T
oken.category==CD}{Token.category==".",Token.length==1}{Token.category==CD})
)

Rule: Instancias69
Priority: 25
(
((DECIMAL){Token.category==CD}):sust2
({NPJK.kind=="NPJK"}):sust1
)
-->
{
try {
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString(
);
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias69");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_quantity");
}

outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Instancias70
Input:  Lookup Token
Options: control = appelt

Rule: Instancias70
Priority:50
(
{Lookup.majorType==music_genre}
):inst

-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("inst");
FeatureMap features =Factory.newFeatureMap();
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Instancias70");
features.put("className","music_genre");
features.put("attributeName", "has_name");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(), as.lastNode(),"Instancias",features);
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

C. Anexo: Reglas JAPE para extracción de atributos

```
Phase:Test5
Input:  Token Lookup NPJK
Options: control = appelt

/
*************************************************************************************************
*****************************************************/
Macro:Atrib_appearance
(
{Token.string==~"[Bb]eautiful",Token.category==JJ}|{Token.string==~"[Uu]gly",Token.category==JJ}|{
Token.string==~"[Cc]lean",Token.category==JJ}|{Token.string==~"[Dd]irty",Token.category==JJ}
|{Token.string==~"[Cc]omplex",Token.category==JJ}|{Token.string==~"[Ss]afe",Token.category==JJ}|{T
oken.string==~"[Dd]angerous",Token.category==JJ}|{Token.string==~"[Ss]trong",Token.category==JJ}
|{Token.string==~"[Ww]eak",Token.category==JJ}|{Token.string==~"[Ss]ame",Token.category==JJ}|{Toke
n.string==~"[Nn]eat",Token.category==JJ}|{Token.string==~"[Mm]essy",Token.category==JJ}
|{Token.string==~"[Rr]ich",Token.category==JJ}|{Token.string==~"[Pp]oor",Token.category==JJ}|{Toke
n.string==~"[Aa]morphous",Token.category==JJ}
|{Token.string==~"[Ss]cenic",Token.category==JJ}
)

Rule: Appearance
Priority: 35
(
(Atrib_appearance):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toStrin
g();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString(
);
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Appearance");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_appearance");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test14
Input:  Token Lookup NPJK
Options: control = appelt

/
**************************************************************************************************
*****************************************************/
Macro:Atrib_brightness
(
{Token.string==~"[Dd]ark",Token.category==JJ}|{Token.string==~"[BB]right",Token.category==JJ}|{Tok
en.string==~"[Ss]hadowy",Token.category==JJ}|{Token.string==~"[Dd]rab",Token.category==JJ}
|{Token.string==~"[Rr]adiant",Token.category==JJ}|{Token.string==~"[Ss]hining",Token.category==JJ}
|{Token.string==~"[Pp]ale",Token.category==JJ}|{Token.string==~"[Dd]ull",Token.category==JJ}
|{Token.string==~"[Gg]lowing",Token.category==JJ}|{Token.string==~"[Ss]himmering",Token.category==
JJ}|{Token.string==~"[Ll]uminous",Token.category==JJ}|{Token.string==~"[Gg]leaming",Token.category
==JJ}
)

Rule: Brightness
Priority: 35
(
(Atrib_brightness):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toStrin
g();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString(
);
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Brightness");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_brightness");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test16
Input:  Token Lookup NPJK
Options: control = appelt

/
**************************************************************************************************
*****************************************************/
Macro:Atrib_color
(
{Token.string==~"[Pp]ink",Token.category==JJ}|{Token.string==~"[Rr]ed",Token.category==JJ}|{Token.
string==~"[Oo]range",Token.category==JJ}|{Token.string==~"[Yy]ellowish",Token.category==JJ}
|{Token.string==~"[Dd]ark-
green",Token.category==JJ}|{Token.string==~"[Bb]lue",Token.category==JJ}|{Token.string==~"[Pp]urpl
e",Token.category==JJ}|{Token.string==~"[Bb]lack",Token.category==JJ}
|{Token.string==~"[Ww]hite",Token.category==JJ}|{Token.string==~"[Gg]ray",Token.category==JJ}|{Tok
en.string==~"[Bb]rown",Token.category==JJ}|{Token.string==~"[Tt]anned",Token.category==JJ}
|{Token.string==~"[Pp]astel",Token.category==JJ}|{Token.string==~"[Mm]etallic",Token.category==JJ}
|{Token.string==~"[Ss]ilver",Token.category==JJ}|{Token.string==~"[Cc]olorless",Token.category==JJ
}
|{Token.string==~"[Tt]ransparent",Token.category==JJ}|{Token.string==~"[Tt]ranslucent",Token.categ
ory==JJ}
)

Rule: Color
Priority: 35
(
(Atrib_color|{Lookup.majorType==color}):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toStrin
g();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString(
);
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Color");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_color");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test13
Input:  Token Lookup NPJK
Options: control = appelt

/
**************************************************************************************************
*****************************************************/
Macro:Atrib_condition
(
{Token.string==~"[Cc]razy",Token.category==JJ}|{Token.string==~"[Ss]ane",Token.category==JJ}|{Toke
n.string==~"[Ss]ick",Token.category==JJ}|{Token.string==~"[Hh]ealthy",Token.category==JJ}
|{Token.string==~"[Dd]runk",Token.category==JJ}|{Token.string==~"[Ss]ober",Token.category==JJ}|{To
ken.string==~"[Tt]ired",Token.category==JJ}|{Token.string==~"[Bb]roken",Token.category==JJ}
|{Token.string==~"[Dd]ead",Token.category==JJ}|{Token.string==~"[Aa]live",Token.category==JJ}
|{Token.string==~"[Hh]ungry",Token.category==JJ}|{Token.string==~"[Aa]sleep",Token.category==JJ}|{
Token.string==~"[Aa]wake",Token.category==JJ}|{Token.string==~"[Bb]usy",Token.category==JJ}
|{Token.string==~"[Ii]dle",Token.category==JJ}|{Token.string==~"[Oo]pen",Token.category==JJ}|{Toke
n.string==~"[Cc]losed",Token.category==JJ}
|{Token.string==~"[Mm]arried",Token.category==JJ}|{Token.string==~"[Ee]ngaged",Token.category==JJ}
|{Token.string==~"[Ss]eparated",Token.category==JJ}|{Token.string==~"[Dd]ivorced",Token.category==
JJ}
|{Token.string==~"[Mm]ixed",Token.category==JJ}|{Token.string==~"[Rr]egulatory",Token.category==JJ
}|{Token.string==~"[Dd]isruptive",Token.category==JJ}|{Token.string==~"[Tt]errorist",Token.categor
y==JJ}
|{Token.string==~"[Bb]etter",Token.category==JJ}|{Token.string==~"[Cc]areful",Token.category==JJ}|
{Token.string==~"[Cc]lever",Token.category==JJ}|{Token.string==~"[Ff]amous",Token.category==JJ}
|{Token.string==~"[Gg]ifted",Token.category==JJ}|{Token.string==~"[Ii]nexpensive",Token.category==
JJ}|{Token.string==~"[Mm]ushy",Token.category==JJ}|{Token.string==~"[Pp]owerful",Token.category==J
J}
|{Token.string==~"[Ss]hy",Token.category==JJ}|{Token.string==~"[Tt]ender",Token.category==JJ}|{Tok
en.string==~"[Uu]ninterested",Token.category==JJ}|{Token.string==~"[Ee]xpensive",Token.category==J
J}
|{Token.string==~"[Ll]ive-music",Token.category==JJ}
)

Rule: Condition
Priority: 35
(
(Atrib_condition):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toStrin
g();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString(
);
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Condition");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_condition");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test3
Input:  Token Lookup NPJK
Options: control = appelt

/
**************************************************************************************************
*****************************************************/
Macro:Atrib_cooking
(
{Token.string==~"[Cc]ooked",Token.category==JJ}|{Token.string==~"[Bb]aked",Token.category==JJ}|{To
ken.string==~"[Ff]ried",Token.category==JJ}|{Token.string==~"[Bb]oiled",Token.category==JJ}
|{Token.string==~"[Pp]eeled",Token.category==JJ}|{Token.string==~"[Ss]liced",Token.category==JJ}|{
Token.string==~"[Ss]tewed",Token.category==JJ}|{Token.string==~"[Ss]teamed",Token.category==JJ}
|{Token.string==~"[Rr]oast",Token.category==JJ}|{Token.string==~"[Bb]roiled",Token.category==JJ}|{
Token.string==~"[Cc]ut",Token.category==JJ}|{Token.string==~"[Gg]rated",Token.category==JJ}
)

Rule: Cooking
Priority: 35
(
(Atrib_cooking):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toStrin
g();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString(
);
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Cooking");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_cooking_state");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test30
Input:  Token Lookup NPJK
Options: control = appelt

Macro:Atrib_difficulty
(
{Token.string==~"[Dd]ifficult",Token.category==JJ}|{Token.string==~"[Hh]ard",Token.category==JJ}|{
Token.string==~"[Ee]asy",Token.category==JJ}
|{Token.string==~"[Ss]imple",Token.category==JJ}

)

Rule: Difficulty
Priority: 35
(
(Atrib_difficulty):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toStrin
g();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString(
);
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Difficulty");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_difficulty");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test33
Input:  Token Lookup NPJK
Options: control = appelt

Macro:Atrib_dimension
(
{Token.string==~"[Ll]ong",Token.category==JJ}|{Token.string==~"[Ss]hort",Token.category==JJ}|{Toke
n.string==~"[Ll]ittle",Token.category==JJ}
|{Token.string==~"[Bb]ig",Token.category==JJ}|{Token.string==~"[Ss]mall",Token.category==JJ}|{Toke
n.string==~"[Ll]arge",Token.category==JJ}
|{Token.string==~"[Tt]all",Token.category==JJ}|{Token.string==~"[Ww]ide",Token.category==JJ}|{Toke
n.string==~"[Dd]eep",Token.category==JJ}
|{Token.string==~"[Ff]ourth-quarter",Token.category==JJ}
)

Rule: Dimension
Priority: 35
(
(Atrib_dimension):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toStrin
g();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString(
);
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Dimension");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_dimension");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test22
Input:  Token Lookup NPJK
Options: control = appelt

Macro:Atrib_distance
(
{Token.string==~"[Ff]ar",Token.category==JJ}|{Token.string==~"[Dd]istant",Token.category==JJ}|{Tok
en.string==~"[Nn]earby",Token.category==JJ}|{Token.string==~"[Cc]lose",Token.category==JJ}
|{Token.string==~"[Ff]araway",Token.category==JJ}|{Token.string==~"[Oo]utlying",Token.category==JJ
}|{Token.string==~"[Rr]emote",Token.category==JJ}
|{Token.string==~"[Ff]ar-
flung",Token.category==JJ}|{Token.string==~"[Nn]eighboring",Token.category==JJ}|{Token.string==~"[
Hh]andy",Token.category==JJ}
)

Rule: Distance
Priority: 35
(
(Atrib_distance):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toStrin
g();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString(
);
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Distance");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_distance");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test19
Input:  Token Lookup NPJK
Options: control = appelt

Macro:Atrib_distributive
(
{Token.string==~"[Ee]ach",Token.category==JJ}|{Token.string==~"[Ee]very",Token.category==JJ}|{Toke
n.string==~"[Ee]ither",Token.category==JJ}|{Token.string==~"[Nn]either",Token.category==JJ}

)

Rule: Distributive
Priority: 35
(
(Atrib_distributive):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toStrin
g();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString(
);
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Distributive");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_distribution");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test32
Input:  Token Lookup NPJK
Options: control = appelt

Macro:Atrib_domain
(
{Token.string==~"[Ll]ocal",Token.category==JJ}|{Token.string==~"[Gg]eneral",Token.category==JJ}|{T
oken.string==~"[Nn]ational",Token.category==JJ}
|{Token.string==~"[Ss]ocial",Token.category==JJ}|{Token.string==~"[Ii]nternational",Token.category
==JJ}|{Token.string==~"[Rr]egional",Token.category==JJ}
|{Token.string==~"[Oo]ceanic",Token.category==JJ}|{Token.string==~"[Ll]unar",Token.category==JJ}|{
Token.string==~"[Pp]olar",Token.category==JJ}
|{Token.string==~"[Ee]quatorial",Token.category==JJ}|{Token.string==~"[Dd]omestic",Token.category=
=JJ}|{Token.string==~"[Aa]cademic",Token.category==JJ}
|{Token.string==~"[Ii]nstitutional",Token.category==JJ}
)

Rule: Domain
Priority: 35
(
(Atrib_domain):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toStrin
g();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString(
);
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Domain");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_domain");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test6
Input:  Token Lookup NPJK
Options: control = appelt

/
*************************************************************************************************************************************************
*****************************/
Macro:Atrib_feeling
(
{Token.string==~"[Ww]orried",Token.category==JJ}|{Token.string==~"[Aa]ngry",Token.category==JJ}|{Token.string==~"[Ff]rustrated
",Token.category==JJ}|{Token.string==~"[Dd]istressed",Token.category==JJ}
|{Token.string==~"[Dd]isappointed",Token.category==JJ}|{Token.string==~"[Uu]pset",Token.category==JJ}|{Token.string==~"[Dd]epr
essed",Token.category==JJ}|{Token.string==~"[Cc]onfused",Token.category==JJ}
|{Token.string==~"[Bb]ored",Token.category==JJ}|{Token.string==~"[Aa]shamed",Token.category==JJ}|{Token.string==~"[Aa]nnoyed",
Token.category==JJ}|{Token.string==~"[Aa]fraid",Token.category==JJ}
|{Token.string==~"[Hh]opeful",Token.category==JJ}|{Token.string==~"[Tt]hrilled",Token.category==JJ}|{Token.string==~"[Gg]ratef
ul",Token.category==JJ}|{Token.string==~"[Pp]leased",Token.category==JJ}
|{Token.string==~"[Ee]nergetic",Token.category==JJ}|{Token.string==~"[Ii]nspired",Token.category==JJ}|{Token.string==~"[Ee]nth
usiastic",Token.category==JJ}|{Token.string==~"[Ee]cstatic",Token.category==JJ}
|{Token.string==~"[Ss]atisfied",Token.category==JJ}|{Token.string==~"[Cc]ontent",Token.category==JJ}|{Token.string==~"[Cc]onfi
dent",Token.category==JJ}|{Token.string==~"[Aa]mused",Token.category==JJ}
|{Token.string==~"[Aa]ngry",Token.category==JJ}|{Token.string==~"[BB]ewildered",Token.category==JJ}|{Token.string==~"[Cc]lumsy
",Token.category==JJ}|{Token.string==~"[Dd]efeated",Token.category==JJ}
|{Token.string==~"[Ee]mbarrassed",Token.category==JJ}|{Token.string==~"[Ff]ierce",Token.category==JJ}|{Token.string==~"[Gg]rum
py",Token.category==JJ}|{Token.string==~"[Hh]elpless",Token.category==JJ}
|{Token.string==~"[Ii]tchy",Token.category==JJ}|{Token.string==~"[Ll]azy",Token.category==JJ}|{Token.string==~"[Mm]ysterious",
Token.category==JJ}|{Token.string==~"[Nn]ervous",Token.category==JJ}
|{Token.string==~"[Oo]bnoxious",Token.category==JJ}|{Token.string==~"[Pp]anicky",Token.category==JJ}|{Token.string==~"[Rr]epul
sive",Token.category==JJ}|{Token.string==~"[Ss]cary",Token.category==JJ}
|{Token.string==~"[Aa]greeable",Token.category==JJ}|{Token.string==~"[Bb]rave",Token.category==JJ}|{Token.string==~"[Cc]alm",T
oken.category==JJ}|{Token.string==~"[Dd]elightful",Token.category==JJ}
|{Token.string==~"[Ee]ager",Token.category==JJ}|{Token.string==~"[Ff]aithful",Token.category==JJ}|{Token.string==~"[Gg]entle",
Token.category==JJ}|{Token.string==~"[Jj]olly",Token.category==JJ}
|{Token.string==~"[Kk]ind",Token.category==JJ}|{Token.string==~"[Ll]ively",Token.category==JJ}|{Token.string==~"[Oo]bedient",T
oken.category==JJ}|{Token.string==~"[Pp]roud",Token.category==JJ}
|{Token.string==~"[Rr]elieved",Token.category==JJ}|{Token.string==~"[Ss]illy",Token.category==JJ}|{Token.string==~"[Tt]hankful
",Token.category==JJ}|{Token.string==~"[Ww]itty",Token.category==JJ}
|{Token.string==~"[Ll]ovable",Token.category==JJ}
)

Rule: Feelings
Priority: 35
(
(Atrib_feeling):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase = doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName = doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Feelings");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_feeling");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test34
Input:  Token Lookup NPJK
Options: control = appelt

Macro:Atrib_justification
(
{Token.string==~"[Oo]dd",Token.category==JJ}|{Token.string==~"[Ww]rong",Token.category==JJ}|{Token
.string==~"[Rr]ight",Token.category==JJ}
|{Token.string==~"[Ff]air",Token.category==JJ}|{Token.string==~"[Pp]roper",Token.category==JJ}
)

Rule: Justification
Priority: 35
(
(Atrib_justification):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toStrin
g();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString(
);
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Justification");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_justification");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test11
Input:  Token Lookup NPJK
Options: control = appelt

/
*************************************************************************************************
*****************************************************/
Macro:Atrib_material
(
{Token.string==~"[Ii]ron",Token.category==JJ}|{Token.string==~"[Ss]teel",Token.category==JJ}|{Toke
n.string==~"[Rr]ubber",Token.category==JJ}|{Token.string==~"[Pp]aper",Token.category==JJ}
|{Token.string==~"[Ww]ooden",Token.category==JJ}|{Token.string==~"[Pp]lastic",Token.category==JJ}|
{Token.string==~"[Ss]tone",Token.category==JJ}
|{Token.string==~"[Gg]lass",Token.category==JJ}|{Token.string==~"[Ll]eather",Token.category==JJ}|{
Token.string==~"[Ss]ilver",Token.category==JJ}|{Token.string==~"[Gg]old",Token.category==JJ}
|{Token.string==~"[Tt]in",Token.category==JJ}|{Token.string==~"[Cc]otton",Token.category==JJ}|{Tok
en.string==~"[Mm]etal",Token.category==JJ}|{Token.string==~"[Nn]on-metallic",Token.category==JJ}
|{Token.string==~"[Cc]loth",Token.category==JJ}|{Token.string==~"[Cc]oncrete",Token.category==JJ}|
{Token.string==~"[Ff]abric",Token.category==JJ}|{Token.string==~"[Cc]eramic",Token.category==JJ}
|{Token.string==~"[Cc]eramic",Token.category==JJ}|{Token.string==~"[Ss]ilicon",Token.category==JJ}
|{Token.string==~"[Ee]xplosive",Token.category==JJ}
|{Token.string==~"[Ss]ole",Token.category==JJ}
)

Rule: Materials
Priority: 35
(
(Atrib_material):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toStrin
g();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString(
);
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Materials");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_material");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test11
Input:  Token Lookup NPJK
Options: control = appelt

/
**************************************************************************************************
*****************************************************/
Macro:Atrib_material
(
{Token.string==~"[Ii]ron",Token.category==JJ}|{Token.string==~"[Ss]teel",Token.category==JJ}|{Toke
n.string==~"[Rr]ubber",Token.category==JJ}|{Token.string==~"[Pp]aper",Token.category==JJ}
|{Token.string==~"[Ww]ooden",Token.category==JJ}|{Token.string==~"[Pp]lastic",Token.category==JJ}|
{Token.string==~"[Ss]tone",Token.category==JJ}
|{Token.string==~"[Gg]lass",Token.category==JJ}|{Token.string==~"[Ll]eather",Token.category==JJ}|{
Token.string==~"[Ss]ilver",Token.category==JJ}|{Token.string==~"[Gg]old",Token.category==JJ}
|{Token.string==~"[Tt]in",Token.category==JJ}|{Token.string==~"[Cc]otton",Token.category==JJ}|{Tok
en.string==~"[Mm]etal",Token.category==JJ}|{Token.string==~"[Nn]on-metallic",Token.category==JJ}
|{Token.string==~"[Cc]loth",Token.category==JJ}|{Token.string==~"[Cc]oncrete",Token.category==JJ}|
{Token.string==~"[Ff]abric",Token.category==JJ}|{Token.string==~"[Cc]eramic",Token.category==JJ}
|{Token.string==~"[Cc]eramic",Token.category==JJ}|{Token.string==~"[Ss]ilicon",Token.category==JJ}
|{Token.string==~"[Ee]xplosive",Token.category==JJ}
|{Token.string==~"[Ss]ole",Token.category==JJ}
)

Rule: Materials
Priority: 35
(
(Atrib_material):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toStrin
g();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString(
);
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Materials");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_material");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test21
Input:  Token Lookup NPJK
Options: control = appelt

Macro:Atrib_opinion
(
{Token.string==~"[Bb]est",Token.category==JJ}|{Token.string==~"[Ww]orse",Token.category==JJ}|{Toke
n.string==~"[Ww]orst",Token.category==JJ}|{Token.string==~"[Ww]onderful",Token.category==JJ}
|{Token.string==~"[Ss]plendid",Token.category==JJ}|{Token.string==~"[Mm]ediocre",Token.category==J
J}|{Token.string==~"[Aa]wful",Token.category==JJ}|{Token.string==~"[Ff]antastic",Token.category==J
J}
|{Token.string==~"[Pp]retty",Token.category==JJ}|{Token.string==~"[Ww]asteful",Token.category==JJ}
|{Token.string==~"[Cc]omfortable",Token.category==JJ}|{Token.string==~"[Uu]ncomfortable",Token.cat
egory==JJ}
|{Token.string==~"[Vv]aluable",Token.category==JJ}|{Token.string==~"[Ww]orthy",Token.category==JJ}
|{Token.string==~"[Ww]orthless",Token.category==JJ}
|{Token.string==~"[Uu]seful",Token.category==JJ}|{Token.string==~"[Uu]seless",Token.category==JJ}|
{Token.string==~"[Ee]vil",Token.category==JJ}
|{Token.string==~"[Aa]ngelic",Token.category==JJ}|{Token.string==~"[Rr]are",Token.category==JJ}|{T
oken.string==~"[Ss]carce",Token.category==JJ}|{Token.string==~"[Dd]isgusting",Token.category==JJ}
|{Token.string==~"[Aa]mazing",Token.category==JJ}|{Token.string==~"[Ss]urprising",Token.category==
JJ}|{Token.string==~"[Pp]ointless",Token.category==JJ}|{Token.string==~"[Pp]ertinent",Token.catego
ry==JJ}
|{Token.string==~"[Pp]arty-on",Token.category==JJ}
)

Rule: Opinion
Priority: 35
(
(Atrib_opinion):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toStrin
g();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString(
);
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Opinion");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_opinion");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test14
Input:  Token Lookup NPJK
Options: control = appelt

/
**************************************************************************************************
*****************************************************/
Macro:Atrib_origin
(
{Token.string==~"[Ee]uropean",Token.category==JJ}|{Token.string==~"[Ll]atin",Token.category==JJ}|{
Token.string==~"[Gg]reek",Token.category==JJ}|{Token.string==~"[Ff]rench",Token.category==JJ}
|{Token.string==~"[Aa]merican",Token.category==JJ}|{Token.string==~"[Gg]erman",Token.category==JJ}
|{Token.string==~"[Cc]olombian",Token.category==JJ}|{Token.string==~"[Ss]panish",Token.category==J
J}
|{Token.string==~"[Ii]talian",Token.category==JJ}
|{Token.string==~"[Rr]ussian",Token.category==JJ}|{Token.string==~"[Kk]orean",Token.category==JJ}|
{Token.string==~"[Cc]hinese",Token.category==JJ}|{Token.string==~"[Jj]apanese",Token.category==JJ}
|{Token.string==~"[Dd]utch",Token.category==JJ}|{Token.string==~"[Aa]rgentine",Token.category==JJ}
|{Token.string==~"[Mm]exican",Token.category==JJ}|{Token.string==~"[Aa]ustralian",Token.category==
JJ}
|{Token.string==~"[Cc]anadian",Token.category==JJ}|{Token.string==~"[Ee]nglish",Token.category==JJ
}|{Token.string==~"[Ii]rish",Token.category==JJ}
|{Token.string==~"[Ff]loridian",Token.category==JJ}|{Token.string==~"[Nn]orthern",Token.category==
JJ}
)

Rule: Origin
Priority: 35
(
(Atrib_origin|{Lookup.minorType==COUNTRYADJ}):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toStrin
g();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString(
);
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Origin");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_origin");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test4
Input:  Token Lookup NPJK
Options: control = appelt

/
*************************************************************************************************************************************************************
**********************/
Macro:Atrib_personality
(
{Token.string==~"[Hh]onest",Token.category==JJ}|{Token.string==~"[Cc]ourageous",Token.category==JJ}|{Token.string==~"[Oo]ptimistic
",Token.category==JJ}|{Token.string==~"[Ii]ntelligent",Token.category==JJ}
|{Token.string==~"[Ss]incere",Token.category==JJ}|{Token.string==~"[Aa]mbitious
",Token.category==JJ}|{Token.string==~"[Mm]odest",Token.category==JJ}|{Token.string==~"[Ss]ensible",Token.category==JJ}
|{Token.string==~"[Ff]riendly",Token.category==JJ}|{Token.string==~"[Pp]ractical",Token.category==JJ}|{Token.string==~"[Cc]onsidera
te",Token.category==JJ}|{Token.string==~"[Tt]olerant",Token.category==JJ}
|{Token.string==~"[Rr]esponsible",Token.category==JJ}|{Token.string==~"[Gg]enerous",Token.category==JJ}|{Token.string==~"[Dd]iscipl
ined",Token.category==JJ}
|{Token.string==~"[Hh]umorous",Token.category==JJ}|{Token.string==~"[Ss]ympathetic",Token.category==JJ}|{Token.string==~"[Dd]ishone
st",Token.category==JJ}|{Token.string==~"[Pp]essimistic",Token.category==JJ}
|{Token.string==~"[Mm]iserly",Token.category==JJ}|{Token.string==~"[Cc]oward",Token.category==JJ}|{Token.string==~"[Ss]elfish",Toke
n.category==JJ}|{Token.string==~"[Ii]mpatient",Token.category==JJ}|{Token.string==~"[Pp]atient",Token.category==JJ}
|{Token.string==~"[Ll]azy",Token.category==JJ}|{Token.string==~"[Gg]reedy",Token.category==JJ}|{Token.string==~"[Rr]esentful",Token
.category==JJ}|{Token.string==~"[Ee]nvious",Token.category==JJ}
|{Token.string==~"[Jj]ealous",Token.category==JJ}|{Token.string==~"[Pp]ossessive",Token.category==JJ}|{Token.string==~"[Cc]onceited
",Token.category==JJ}|{Token.string==~"[Aa]rrogant",Token.category==JJ}
|{Token.string==~"[Ff]ussy",Token.category==JJ}|{Token.string==~"[Gg]ullible",Token.category==JJ}|{Token.string==~"[Ss]tubborn",Tok
en.category==JJ}|{Token.string==~"[Cc]areless",Token.category==JJ}
|{Token.string==~"[Hh]appy",Token.category==JJ}|{Token.string==~"[Ss]ad",Token.category==JJ}|{Token.string==~"[Ee]xcited",Token.cat
egory==JJ}|{Token.string==~"[Ss]cared",Token.category==JJ}
|{Token.string==~"[Ff]rightened",Token.category==JJ}|{Token.string==~"[Oo]utgoing",Token.category==JJ}|{Token.string==~"[Zz]any",To
ken.category==JJ}|{Token.string==~"[Gg]rumpy",Token.category==JJ}
|{Token.string==~"[Cc]heerful",Token.category==JJ}|{Token.string==~"[Jj]olly",Token.category==JJ}|{Token.string==~"[Cc]arefree",Tok
en.category==JJ}|{Token.string==~"[Qq]uick-witted",Token.category==JJ}
|{Token.string==~"[Bb]lissful",Token.category==JJ}|{Token.string==~"[Ll]onely",Token.category==JJ}|{Token.string==~"[Ee]lated",Toke
n.category==JJ}|{Token.string==~"[Aa]utonomous",Token.category==JJ}
)

Rule: Personality
Priority: 35
(
(Atrib_personality):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase = doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName = doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Personality");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_personality");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test31
Input:  Token Lookup NPJK
Options: control = appelt

Macro:Atrib_primacy
(
{Token.string==~"[Oo]nly",Token.category==JJ}|{Token.string==~"[Pp]articular",Token.category==JJ}|
{Token.string==~"[Mm]ain",Token.category==JJ}
|{Token.string==~"[Ss]pecial",Token.category==JJ}|{Token.string==~"[Mm]ajor",Token.category==JJ}

)

Rule: Primacy
Priority: 35
(
(Atrib_primacy):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toStrin
g();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString(
);
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Primacy");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_primacy");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test29
Input:  Token Lookup NPJK
Options: control = appelt

Macro:Atrib_purpose
(
{Token.string==~"[Ff]olding",Token.category==JJ}|{Token.string==~"[Ss]wing",Token.category==JJ}|{T
oken.string==~"[Ww]ork",Token.category==JJ}
|{Token.string==~"[Rr]acing",Token.category==JJ}|{Token.string==~"[Cc]ooking",Token.category==JJ}|
{Token.string==~"[Ss]leeping",Token.category==JJ}
|{Token.string==~"[Dd]ance",Token.category==JJ}|{Token.string==~"[Rr]olling",Token.category==JJ}|{
Token.string==~"[Ww]alking",Token.category==JJ}
|{Token.string==~"[Kk]nowing",Token.category==JJ}|{Token.string==~"[Dd]eceiving",Token.category==J
J}|{Token.string==~"[Bb]inding",Token.category==JJ}
)

Rule: Purpose
Priority: 35
(
(Atrib_purpose):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toStrin
g();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString(
);
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Purpose");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_purpose");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test37
Input:  Token Lookup NPJK
Options: control = appelt

Macro:Atrib_quality
(
{Token.string=~"[Ll]ight",Token.category==JJ}|{Token.string=~"[Hh]elpful",Token.category==JJ}|{Token.string=~"[Bb]loody",Token.category==JJ}
|{Token.string=~"[Hh]uman",Token.category==JJ}|{Token.string=~"[Ss]orry",Token.category==JJ}|{Token.string=~"[Ss]ure",Token.category==JJ}
|{Token.string=~"[Aa]ble",Token.category==JJ}|{Token.string=~"[Cc]ertain",Token.category==JJ}|{Token.string=~"[Cc]lear",Token.category==JJ}
|{Token.string=~"[Ff]ree",Token.category==JJ}|{Token.string=~"[Nn]ext",Token.category==JJ}|{Token.string=~"[Tt]rue",Token.category==JJ}
|{Token.string=~"[Pp]ossible",Token.category==JJ}|{Token.string=~"[Uu]sual",Token.category==JJ}|{Token.string=~"[Cc]orrect",Token.category==JJ}
|{Token.string=~"[Uu]nusual",Token.category==JJ}|{Token.string=~"[Ss]pecific",Token.category==JJ}|{Token.string=~"[Ll]egislated",Token.category==JJ}
|{Token.string=~"[Ee]xplicit",Token.category==JJ}|{Token.string=~"[Ii]rregular",Token.category==JJ}|{Token.string=~"[Rr]egular",Token.category==JJ}
|{Token.string=~"[Ss]ubject",Token.category==JJ}|{Token.string=~"[Ss]tandard",Token.category==JJ}|{Token.string=~"[Uu]nscreened",Token.category==JJ}
|{Token.string=~"[Cc]ommercial",Token.category==JJ}|{Token.string=~"[Ss]trategic",Token.category==JJ}|{Token.string=~"[Ff]inal",Token.category==JJ}
|{Token.string=~"[Cc]onsultative",Token.category==JJ}|{Token.string=~"[Dd]irect",Token.category==JJ}|{Token.string=~"[Rr]esultant",Token.category==JJ}
|{Token.string=~"[Ss]treamlined",Token.category==JJ}|{Token.string=~"[Ss]ystemic",Token.category==JJ}|{Token.string=~"[Oo]vert",Token.category==JJ}
|{Token.string=~"[Cc]onsistent",Token.category==JJ}|{Token.string=~"[Rr]adical",Token.category==JJ}|{Token.string=~"[Bb]asic",Token.category==JJ}
|{Token.string=~"[Ll]ikely",Token.category==JJ}|{Token.string=~"[Ee]xtensive",Token.category==JJ}|{Token.string=~"[Dd]ecentralized",Token.category==JJ
}
|{Token.string=~"[Uu]nnecessary",Token.category==JJ}|{Token.string=~"[Ee]fficient",Token.category==JJ}|{Token.string=~"[Aa]dequate",Token.category==JJ}
|{Token.string=~"[Oo]riginal",Token.category==JJ}|{Token.string=~"[Cc]ostly",Token.category==JJ}|{Token.string=~"[Oo]verseas",Token.category==JJ}
|{Token.string=~"[Aa]dvisory",Token.category==JJ}|{Token.string=~"[Ll]egislative",Token.category==JJ}|{Token.string=~"[Aa]uthorized",Token.category==J
J}
|{Token.string=~"[Ll]ayered",Token.category==JJ}|{Token.string=~"[Uu]nlawful",Token.category==JJ}
|{Token.string=~"[Pp]ublic",Token.category==JJ}|{Token.string=~"[Dd]ignified",Token.category==JJ}|{Token.string=~"[Rr]equired",Token.category==JJ}
|{Token.string=~"[Pp]rojected",Token.category==JJ}|{Token.string=~"[Dd]egradable",Token.category==JJ}|{Token.string=~"[Bb]lood-
sample",Token.category==JJ}
|{Token.string=~"[Rr]easonable",Token.category==JJ}|{Token.string=~"[Aa]lternative",Token.category==JJ}|{Token.string=~"[Dd]riftless",Token.category==
JJ}
|{Token.string=~"[Oo]ptimum",Token.category==JJ}|{Token.string=~"[Rr]espective",Token.category==JJ}|{Token.string=~"[Rr]andom",Token.category==JJ}
|{Token.string=~"[Rr]elative",Token.category==JJ}|{Token.string=~"[Ss]oil-test",Token.category==JJ}|{Token.string=~"[Mm]ean",Token.category==JJ}
|{Token.string=~"[Tt]raditional",Token.category==JJ}|{Token.string=~"[Ss]eedbed",Token.category==JJ}|{Token.string=~"[Cc]orn-
based",Token.category==JJ}
|{Token.string=~"[Ss]plit-plot",Token.category==JJ}|{Token.string=~"[Ll]ong-
term",Token.category==JJ}|{Token.string=~"[Ss]ubsequent",Token.category==JJ}
|{Token.string=~"[Pp]rivate",Token.category==JJ}|{Token.string=~"[Gg]enetic",Token.category==JJ}|{Token.string=~"[Ss]hort-term",Token.category==JJ}
|{Token.string=~"[Cc]ontinuous",Token.category==JJ}|{Token.string=~"[Ii]nherent",Token.category==JJ}|{Token.string=~"[Ii]ndependent",Token.category==J
J}
|{Token.string=~"[Ee]conomic",Token.category==JJ}|{Token.string=~"[Kk]ey",Token.category==JJ}|{Token.string=~"[Cc]rop-input",Token.category==JJ}
|{Token.string=~"[Ff]avorable",Token.category==JJ}|{Token.string=~"[Oo]n-farm",Token.category==JJ}|{Token.string=~"[Mm]ental",Token.category==JJ}
|{Token.string=~"[Pp]redictive",Token.category==JJ}|{Token.string=~"[Pp]revious",Token.category==JJ}|{Token.string=~"[Cc]ognitive",Token.category==JJ}
|{Token.string=~"[Ee]ffective",Token.category==JJ}|{Token.string=~"[Pp]rofessional",Token.category==JJ}|{Token.string=~"[Mm]aladaptive",Token.category
==JJ}
|{Token.string=~"[Ss]elf-talk",Token.category==JJ}|{Token.string=~"[Aa]verage",Token.category==JJ}|{Token.string=~"[Dd]eaf",Token.category==JJ}
|{Token.string=~"[Hh]oarse",Token.category==JJ}|{Token.string=~"[Pp]ositive",Token.category==JJ}|{Token.string=~"[Rr]elevant",Token.category==JJ}
|{Token.string=~"[Nn]egative",Token.category==JJ}|{Token.string=~"[Aa]ppropriate",Token.category==JJ}|{Token.string=~"[Uu]nknown",Token.category==JJ}
|{Token.string=~"[Cc]heap",Token.category==JJ}|{Token.string=~"[Ff]alse",Token.category==JJ}|{Token.string=~"[Aa]ccurate",Token.category==JJ}
|{Token.string=~"[Cc]omplete",Token.category==JJ}|{Token.string=~"[Aa]thletic",Token.category==JJ}|{Token.string=~"[Pp]ure",Token.category==JJ}
|{Token.string=~"[Ww]rinkly",Token.category==JJ}|{Token.string=~"[Ww]orldly",Token.category==JJ}|{Token.string=~"[Ww]ise",Token.category==JJ}
|{Token.string=~"[Ww]illful",Token.category==JJ}|{Token.string=~"[Ww]ild",Token.category==JJ}|{Token.string=~"[Vv]iolent",Token.category==JJ}
|{Token.string=~"[Vv]icious",Token.category==JJ}|{Token.string=~"[Uu]ttar",Token.category==JJ}|{Token.string=~"[Uu]nfortunate",Token.category==JJ}
|{Token.string=~"[Tt]riumphant",Token.category==JJ}|{Token.string=~"[Tt]remendous",Token.category==JJ}|{Token.string=~"[Tt]houghtful",Token.category==
JJ}
|{Token.string=~"[Tt]horough",Token.category==JJ}|{Token.string=~"[Tt]errible",Token.category==JJ}|{Token.string=~"[Ss]uspicious",Token.category==JJ}
|{Token.string=~"[Ss]urly",Token.category==JJ}|{Token.string=~"[Ss]uccessful",Token.category==JJ}|{Token.string=~"[Ss]trict",Token.category==JJ}
|{Token.string=~"[Ss]mart",Token.category==JJ}|{Token.string=~"[Ss]heepish",Token.category==JJ}|{Token.string=~"[Ss]edate",Token.category==JJ}
|{Token.string=~"[Ss]earching",Token.category==JJ}|{Token.string=~"[Rr]ipe",Token.category==JJ}|{Token.string=~"[Rr]igid",Token.category==JJ}
|{Token.string=~"[Rr]ightful",Token.category==JJ}|{Token.string=~"[Rr]ighteous",Token.category==JJ}|{Token.string=~"[Rr]eatful",Token.category==JJ}
|{Token.string=~"[Rr]eproachful",Token.category==JJ}|{Token.string=~"[Rr]eluctant",Token.category==JJ}|{Token.string=~"[Rr]eassuring",Token.category==
JJ}
|{Token.string=~"[Rr]eady",Token.category==JJ}|{Token.string=~"[Rr]avenous",Token.category==JJ}|{Token.string=~"[Qq]uintessential",Token.category==JJ}
|{Token.string=~"[Qq]ueenly",Token.category==JJ}|{Token.string=~"[Pp]otential",Token.category==JJ}|{Token.string=~"[Pp]layful",Token.category==JJ}
|{Token.string=~"[Pp]artial",Token.category==JJ}|{Token.string=~"[Oo]rderly",Token.category==JJ}|{Token.string=~"[Nn]ear",Token.category==JJ}
|{Token.string=~"[Nn]atural",Token.category==JJ}|{Token.string=~"[Mm]ocking",Token.category==JJ}|{Token.string=~"[Mm]iserable",Token.category==JJ}
|{Token.string=~"[Mm]eaningful",Token.category==JJ}|{Token.string=~"[Mm]ad",Token.category==JJ}|{Token.string=~"[Kk]nightly",Token.category==JJ}
|{Token.string=~"[Kk]ingly",Token.category==JJ}|{Token.string=~"[Ii]nward",Token.category==JJ}|{Token.string=~"[Ii]ntense",Token.category==JJ}
|{Token.string=~"[Ii]ntent",Token.category==JJ}|{Token.string=~"[Ii]natant",Token.category==JJ}|{Token.string=~"[Ii]nnocent",Token.category==JJ}
|{Token.string=~"[Ii]ncredible",Token.category==JJ}|{Token.string=~"[Ii]mmediate",Token.category==JJ}|{Token.string=~"[Hh]opeless",Token.category==JJ}
|{Token.string=~"[Hh]oly",Token.category==JJ}|{Token.string=~"[Gg]leeful",Token.category==JJ}|{Token.string=~"[Gg]hostly",Token.category==JJ}
|{Token.string=~"[Gg]hastly",Token.category==JJ}|{Token.string=~"[Gg]arden",Token.category==JJ}|{Token.string=~"[Ff]urious",Token.category==JJ}
|{Token.string=~"[Ff]rightful",Token.category==JJ}|{Token.string=~"[Ff]renetic",Token.category==JJ}|{Token.string=~"[Ff]rantic",Token.category==JJ}
|{Token.string=~"[Ff]rank",Token.category==JJ}|{Token.string=~"[Ff]oolish",Token.category==JJ}|{Token.string=~"[Ff]ervent",Token.category==JJ}
|{Token.string=~"[Ff]erocious",Token.category==JJ}|{Token.string=~"[Ee]xtreme",Token.category==JJ}|{Token.string=~"[Ee]xcellent",Token.category==JJ}
|{Token.string=~"[Ee]xact",Token.category==JJ}|{Token.string=~"[Ee]normous",Token.category==JJ}|{Token.string=~"[Dd]reamy",Token.category==JJ}
|{Token.string=~"[Dd]iligent",Token.category==JJ}|{Token.string=~"[Dd]esperate",Token.category==JJ}|{Token.string=~"[Dd]ear",Token.category==JJ}
|{Token.string=~"[Cc]urious",Token.category==JJ}|{Token.string=~"[Cc]rosa",Token.category==JJ}|{Token.string=~"[Cc]ontinual",Token.category==JJ}
|{Token.string=~"[Cc]ommon",Token.category==JJ}|{Token.string=~"[Cc]oaxing",Token.category==JJ}|{Token.string=~"[Bb]risk",Token.category==JJ}
|{Token.string=~"[Bb]old",Token.category==JJ}|{Token.string=~"[Bb]ashful",Token.category==JJ}|{Token.string=~"[Aa]nxious",Token.category==JJ}
|{Token.string=~"[Aa]ffectionate",Token.category==JJ}|{Token.string=~"[Aa]bsentminded",Token.category==JJ}|{Token.string=~"[Ss]elf-
propelled",Token.category==JJ}
|{Token.string=~"[Pp]arental",Token.category==JJ}|{Token.string=~"[Ff]iscal",Token.category==JJ}
)


Rule: Qualification
Priority: 50
(
(Atrib_quality):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase = doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName = doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Qualification");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_qualification");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test17
Input:  Token Lookup NPJK
Options: control = appelt

/
*************************************************************************************************
*****************************************************/
Macro:Atrib_quantity
(
{Token.string==~"[Mm]uch",Token.category==JJ}|{Token.string==~"[Aa]ny",Token.category==JJ}|{Token.
string==~"[Ff]ew",Token.category==JJ}|{Token.string==~"[Ss]everal",Token.category==JJ}
|{Token.string==~"[Mm]ost",Token.category==JJ}|{Token.string==~"[Aa]ll",Token.category==JJ}|{Token
.string==~"[Ee]nough",Token.category==JJ}
|{Token.string==~"[Ss]ufficient",Token.category==JJ}|{Token.string==~"[Aa]bundant",Token.category=
=JJ}|{Token.string==~"[Ee]mpty",Token.category==JJ}
|{Token.string==~"[Hh]eavy",Token.category==JJ}|{Token.string==~"[Nn]umerous",Token.category==JJ}
|{Token.string==~"[Dd]ouble",Token.category==JJ}|{Token.string==~"[Ss]parse",Token.category==JJ}|{
Token.string==~"[Ss]ubstantial",Token.category==JJ}|{Token.string==~"[Hh]alf",Token.category==JJ}
|{Token.string==~"[Nn]o",Token.category==JJ}|{Token.string==~"[Ii]nsufficient",Token.category==JJ}
|{Token.string==~"[Mm]ore",Token.category==JJ}|{Token.string==~"[Pp]lenty",Token.category==JJ}
|{Token.string==~"[Mm]ajority",Token.category==JJ}|{Token.string==~"[Rr]est",Token.category==JJ}|{
Token.string==~"[Mm]ultiple",Token.category==JJ}|{Token.string==~"[Tt]riple",Token.category==JJ}
|{Token.string==~"[Qq]uantitative",Token.category==JJ}|{Token.string==~"[Mm]inimum",Token.category
==JJ}|{Token.string==~"[Oo]verall",Token.category==JJ}|{Token.string==~"[Rr]obust",Token.category=
=JJ}
|{Token.string==~"[Mm]ultilateral",Token.category==JJ}|{Token.string==~"[Ff]ull",Token.category==J
J}|{Token.string==~"[Ss]ingle",Token.category==JJ}
|{Token.string==~"[Ww]hole",Token.category==JJ}|{Token.string==~"[Ss]ome",Token.category==JJ}|{Tok
en.string==~"[Mm]any",Token.category==JJ}|{Token.string==~"[Ee]xtra",Token.category==JJ}
|{Token.string==~"[Cc]ountless",Token.category==JJ}
)

Rule: Quantity
Priority: 35
(
(Atrib_quantity):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toStrin
g();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString(
);
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Quantity");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_quantity");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test12
Input:  Token Lookup NPJK
Options: control = appelt

/
**************************************************************************************************
*****************************************************/
Macro:Atrib_religion
(
{Token.string==~"[Cc]atholic",Token.category==JJ}|{Token.string==~"[Pp]rotestant",Token.category==
JJ}|{Token.string==~"[Aa]nglican",Token.category==JJ}|{Token.string==~"[Bb]aptist",Token.category=
=JJ}
|{Token.string==~"[Cc]hristian",Token.category==JJ}|{Token.string==~"[Hh]indu",Token.category==JJ}
|{Token.string==~"[Bb]uddhist",Token.category==JJ}|{Token.string==~"[Mm]uslin",Token.category==JJ}
|{Token.string==~"[Jj]ewish",Token.category==JJ}|{Token.string==~"[Ll]utheran",Token.category==JJ}
)

Rule: Religion
Priority: 35
(
(Atrib_religion):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toStrin
g();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString(
);
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Religion");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_religion");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test7
Input:  Token Lookup NPJK
Options: control = appelt

/
*************************************************************************************************
************************************************/
Macro:Atrib_shape
(
{Token.string==~"[Ss]quare",Token.category==JJ}|{Token.string==~"[Rr]ound",Token.category==JJ}|{Token.
string==~"[Rr]ectangular",Token.category==JJ}|{Token.string==~"[Tt]riangular",Token.category==JJ}
|{Token.string==~"[Oo]val",Token.category==JJ}|{Token.string==~"[Cc]onical",Token.category==JJ}|{Token
.string==~"[Ss]pherical",Token.category==JJ}|{Token.string==~"[Cc]ubical",Token.category==JJ}
|{Token.string==~"[Cc]ylindrical",Token.category==JJ}|{Token.string==~"[Ss]traight",Token.category==JJ
}|{Token.string==~"[Cc]urved",Token.category==JJ}|{Token.string==~"[Cc]rooked",Token.category==JJ}
|{Token.string==~"[Ff]lat",Token.category==JJ}|{Token.string==~"[Ss]teep",Token.category==JJ}|{Token.s
tring==~"[Hh]ollow",Token.category==JJ}|{Token.string==~"[Cc]ircular",Token.category==JJ}
|{Token.string==~"[Ss]leek",Token.category==JJ}|{Token.string==~"[Bb]lobby",Token.category==JJ}|{Token
.string==~"[Rr]otund",Token.category==JJ}|{Token.string==~"[Gg]lobular",Token.category==JJ}
|{Token.string==~"[Ww]avy",Token.category==JJ}|{Token.string==~"[Oo]blong",Token.category==JJ}|{Token.
string==~"[Ee]lliptical",Token.category==JJ}|{Token.string==~"[Zz]igzag",Token.category==JJ}
|{Token.string==~"[Ss]quiggly",Token.category==JJ}|{Token.string==~"[Ww]inding",Token.category==JJ}|{T
oken.string==~"[Ss]erpentine",Token.category==JJ}|{Token.string==~"[Ww]arped",Token.category==JJ}
|{Token.string==~"[Dd]istorted",Token.category==JJ}|{Token.string==~"[Bb]road",Token.category==JJ}|{To
ken.string==~"[Ss]kinny",Token.category==JJ}
|{Token.string==~"[Ss]erried",Token.category==JJ}
)

Rule: Shape
Priority: 35
(
(Atrib_shape):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Shape");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_shape");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test35
Input:  Token Lookup NPJK
Options: control = appelt

Macro:Atrib_similarity
(
{Token.string==~"[Oo]ther",Token.category==JJ}|{Token.string==~"[Dd]ifferent",Token.category==JJ}|{Tok
en.string==~"[Ss]imilar",Token.category==JJ}
|{Token.string==~"[Ll]ike",Token.category==JJ}
)

Rule: Similarity
Priority: 35
(
(Atrib_similarity):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Similarity");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_similarity");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test1
Input:  Token Lookup NPJK
Options: control = appelt

Macro:Atrib_size
(
{Token.string==~"[Tt]hin",Token.category==JJ}|{Token.string==~"[Tt]hick",Token.category==JJ}|{Token.st
ring==~"[Bb]ig-boned",Token.category==JJ}|{Token.string==~"[Cc]hubby",Token.category==JJ}
|{Token.string==~"[Ff]lat-
boned",Token.category==JJ}|{Token.string==~"[Gg]iant",Token.category==JJ}|{Token.string==~"[Gg]igantic
",Token.category==JJ}|{Token.string==~"[Hh]uge",Token.category==JJ}
|{Token.string==~"[Ii]nmense",Token.category==JJ}|{Token.string==~"[Jj]umbo",Token.category==JJ}|{Toke
n.string==~"[Mm]ajestic",Token.category==JJ}|{Token.string==~"[Mm]ammoth",Token.category==JJ}
|{Token.string==~"[Mm]assive",Token.category==JJ}|{Token.string==~"[Mm]iniature",Token.category==JJ}|{
Token.string==~"[Pp]etite",Token.category==JJ}|{Token.string==~"[Pp]uny",Token.category==JJ}
|{Token.string==~"[Ss]crawny",Token.category==JJ}|{Token.string==~"[Tt]eeny",Token.category==JJ}|{Toke
n.string==~"[Tt]iny",Token.category==JJ}|{Token.string==~"[Vv]ast",Token.category==JJ}
|{Token.string==~"[Mm]edium",Token.category==JJ}|{Token.string==~"[Nn]arrow",Token.category==JJ}|{Toke
n.string==~"[Ss]hallow",Token.category==JJ}|{Token.string==~"[Vv]arious",Token.category==JJ}
|{Token.string==~"[Cc]olossal",Token.category==JJ}|{Token.string==~"[Ff]at",Token.category==JJ}|{Token
.string==~"[Ii]mmense",Token.category==JJ}|{Token.string==~"[Tt]eeny-tinytiny",Token.category==JJ}
)

Rule: Size
Priority: 35
(
(Atrib_size):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Size");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_size");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test25
Input:  Token Lookup NPJK
Options: control = appelt

Macro:Atrib_smell
(
{Token.string==~"[Pp]erfumed",Token.category==JJ}|{Token.string==~"[Aa]crid",Token.category==JJ}|{Toke
n.string==~"[Pp]utrid",Token.category==JJ}|{Token.string==~"[Bb]urnt",Token.category==JJ}
|{Token.string==~"[Ss]melly",Token.category==JJ}|{Token.string==~"[Rr]eeking",Token.category==JJ}|{Tok
en.string==~"[Nn]oxious",Token.category==JJ}|{Token.string==~"[Pp]unget",Token.category==JJ}
|{Token.string==~"[Aa]romatic",Token.category==JJ}|{Token.string==~"[Ff]ragant",Token.category==JJ}|{T
oken.string==~"[Ss]cented",Token.category==JJ}
|{Token.string==~"[Mm]usty",Token.category==JJ}|{Token.string==~"[Ss]weet-
smelling",Token.category==JJ}
)

Rule: Smell
Priority: 35
(
(Atrib_distributive):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Smell");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_smell");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test27
Input:  Token Lookup NPJK
Options: control = appelt

Macro:Atrib_sound
(
{Token.string==~"[Ll]oud",Token.category==JJ}|{Token.string==~"[Ss]ilent",Token.category==JJ}|{Token.s
tring==~"[Vv]ociferous",Token.category==JJ}|{Token.string==~"[Ss]creaming",Token.category==JJ}
|{Token.string==~"[Ss]houting",Token.category==JJ}|{Token.string==~"[Tt]hunderrous",Token.category==JJ
}|{Token.string==~"[Bb]laring",Token.category==JJ}
|{Token.string==~"[Qq]uiet",Token.category==JJ}|{Token.string==~"[Nn]oisy",Token.category==JJ}|{Token.
string==~"[Tt]alkative",Token.category==JJ}
|{Token.string==~"[Rr]owdy",Token.category==JJ}|{Token.string==~"[Dd]eafening",Token.category==JJ}|{To
ken.string==~"[Ff]aint",Token.category==JJ}
|{Token.string==~"[Mm]uffled",Token.category==JJ}|{Token.string==~"[Mm]ute",Token.category==JJ}|{Token
.string==~"[Mm]ute",Token.category==JJ}
|{Token.string==~"[Ss]peechless",Token.category==JJ}|{Token.string==~"[Ww]hispered",Token.category==JJ
}|{Token.string==~"[Hh]ushed",Token.category==JJ}
|{Token.string==~"[Cc]ooing",Token.category==JJ}|{Token.string==~"[Hh]issing",Token.category==JJ}|{Tok
en.string==~"[Mm]elodic",Token.category==JJ}
|{Token.string==~"[Pp]urring",Token.category==JJ}|{Token.string==~"[Rr]aspy",Token.category==JJ}|{Toke
n.string==~"[Ss]creeching",Token.category==JJ}
|{Token.string==~"[Tt]hundering",Token.category==JJ}|{Token.string==~"[Vv]oiceless",Token.category==JJ
}|{Token.string==~"[Ww]hispering",Token.category==JJ}
|{Token.string==~"[Ii]naudible",Token.category==JJ}|{Token.string==~"[Aa]udible",Token.category==JJ}
)

Rule: Sound
Priority: 35
(
(Atrib_sound):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Sound");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_sound");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test24
Input:  Token Lookup NPJK
Options: control = appelt

Macro:Atrib_speed
(
{Token.string==~"[Qq]uick",Token.category==JJ}|{Token.string==~"[Ff]ast",Token.category==JJ}|{Token.st
ring==~"[Rr]apid",Token.category==JJ}|{Token.string==~"[Ss]low",Token.category==JJ}
|{Token.string==~"[Ss]wift",Token.category==JJ}|{Token.string==~"[Ss]peeding",Token.category==JJ}|{Tok
en.string==~"[Rr]ushing",Token.category==JJ}
|{Token.string==~"[Bb]ustling",Token.category==JJ}|{Token.string==~"[Ss]nappy",Token.category==JJ}|{To
ken.string==~"[Ww]hirlwind",Token.category==JJ}
|{Token.string==~"[Hh]asty",Token.category==JJ}|{Token.string==~"[Pp]rompt",Token.category==JJ}|{Token
.string==~"[Bb]rief",Token.category==JJ}
)

Rule: Speed
Priority: 35
(
(Atrib_speed):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Speed");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_speed");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test1
Input:  Token Lookup NPJK
Options: control = appelt

Macro:Atrib_state
(
{Token.string==~"[Ll]iquid",Token.category==JJ}|{Token.string==~"[Ss]olid",Token.category==JJ}|{Token.
string==~"[Gg]as",Token.category==JJ}|{Token.string==~"[Ee]utectic",Token.category==JJ}
|{Token.string==~"[Aa]ssociate",Token.category==JJ}|{Token.string==~"[Cc]ivil",Token.category==JJ}|{To
ken.string==~"[Uu]nstable",Token.category==JJ}|{Token.string==~"[Ss]terile",Token.category==JJ}
|{Token.string==~"[Ii]ntermediate",Token.category==JJ}|{Token.string==~"[Pp]reventive",Token.category=
=JJ}|{Token.string==~"[Oo]ngoing",Token.category==JJ}
|{Token.string==~"[Aa]mino",Token.category==JJ}|{Token.string==~"[Cc]rucial",Token.category==JJ}

)

Rule: State
Priority: 50
(
(Atrib_state):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","State");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_state");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test2
Input:  Token Lookup NPJK
Options: control = appelt

/
***************Sabor*****************************************************************************
*****************************************************/
Macro:Atrib_taste
(
{Token.string==~"[Ss]weet",Token.category==JJ}|{Token.string==~"[Ss]alty",Token.category==JJ}|{Token.s
tring==~"[Ss]our",Token.category==JJ}|{Token.string==~"[Bb]itter",Token.category==JJ}
|{Token.string==~"[Gg]reasy",Token.category==JJ}|{Token.string==~"[Ff]resh",Token.category==JJ}|{Token
.string==~"[Ss]tale",Token.category==JJ}|{Token.string==~"[Tt]asty",Token.category==JJ}
|{Token.string==~"[Dd]elicious",Token.category==JJ}|{Token.string==~"[Tt]asteless",Token.category==JJ}
|{Token.string==~"[Ff]atty",Token.category==JJ}|{Token.string==~"[Rr]otten",Token.category==JJ}
|{Token.string==~"[Ss]picy",Token.category==JJ}|{Token.string==~"[Aa]cidic",Token.category==JJ}|{Token
.string==~"[Ss]avory",Token.category==JJ}|{Token.string==~"[Dd]electable",Token.category==JJ}
|{Token.string==~"[Yy]ummy",Token.category==JJ}|{Token.string==~"[Bb]land",Token.category==JJ}|{Token.
string==~"[Pp]alatable",Token.category==JJ}|{Token.string==~"[Ll]uscious",Token.category==JJ}
|{Token.string==~"[Aa]ppetizing",Token.category==JJ}|{Token.string==~"[Ww]atery",Token.category==JJ}|{
Token.string==~"[Jj]uicy",Token.category==JJ}|{Token.string==~"[Nn]utritious",Token.category==JJ}
|{Token.string==~"[Tt]art",Token.category==JJ}
)

Rule: Taste
Priority: 35
(
(Atrib_taste):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Taste");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_taste");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test23
Input:  Token Lookup NPJK
Options: control = appelt

Macro:Atrib_temperature
(
{Token.string==~"[Hh]ot",Token.category==JJ}|{Token.string==~"[Cc]old",Token.category==JJ}|{Token.stri
ng==~"[Ff]reezing",Token.category==JJ}|{Token.string==~"[Ii]cy",Token.category==JJ}
|{Token.string==~"[Ff]rigid",Token.category==JJ}|{Token.string==~"[Ss]weltering",Token.category==JJ}|{
Token.string==~"[Ww]intry",Token.category==JJ}
|{Token.string==~"[Ff]rosty",Token.category==JJ}|{Token.string==~"[Ff]rozen",Token.category==JJ}|{Toke
n.string==~"[Nn]ippy",Token.category==JJ}|{Token.string==~"[Cc]hilly",Token.category==JJ}
|{Token.string==~"[Ss]izzling",Token.category==JJ}|{Token.string==~"[Ss]calding",Token.category==JJ}|{
Token.string==~"[Bb]urning",Token.category==JJ}
|{Token.string==~"[Ff]everish",Token.category==JJ}|{Token.string==~"[Ff]iery",Token.category==JJ}|{Tok
en.string==~"[Ss]teaming",Token.category==JJ}
|{Token.string==~"[Cc]ool",Token.category==JJ}|{Token.string==~"[Ww]arm",Token.category==JJ}
)

Rule: Temperature
Priority: 35
(
(Atrib_temperature):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Temperature");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_temperature");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test10
Input:  Token Lookup NPJK
Options: control = appelt

/
**************************************************************************************************************
*************************************************/
Macro:Atrib_texture
(
{Token.string==~"[Rr]ough",Token.category==JJ}|{Token.string==~"[Ww]et",Token.category==JJ}|{Token.str
ing==~"[Ss]lippery",Token.category==JJ}
|{Token.string==~"[Ss]ticky",Token.category==JJ}|{Token.string==~"[Ee]ven",Token.category==JJ}
|{Token.string==~"[Ss]harp",Token.category==JJ}|{Token.string==~"[Bb]lunt",Token.category==JJ}
|{Token.string==~"[Tt]ight",Token.category==JJ}|{Token.string==~"[Ll]oose",Token.category==JJ}
)

Rule: Texture
Priority: 35
(
(Atrib_texture):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Texture");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_texture");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test8
Input:  Token Lookup NPJK
Options: control = appelt

/
*************************************************************************************************
************************************************/
Macro:Atrib_time
(
{Token.string==~"[Oo]ld",Token.category==JJ}|{Token.string==~"[Yy]oung",Token.category==JJ}|{Token.str
ing==~"[Nn]ew",Token.category==JJ}|{Token.string==~"[Mm]odern",Token.category==JJ}
|{Token.string==~"[Aa]ncient",Token.category==JJ}|{Token.string==~"[Uu]pdated",Token.category==JJ}|{To
ken.string==~"[Oo]utdated",Token.category==JJ}
|{Token.string==~"[Ss]enior",Token.category==JJ}|{Token.string==~"[Jj]unior",Token.category==JJ}
|{Token.string==~"[Cc]urrent",Token.category==JJ}|{Token.string==~"[Pp]ast",Token.category==JJ}|{Token
.string==~"[Ff]uture",Token.category==JJ}|{Token.string==~"[Yy]early",Token.category==JJ}
|{Token.string==~"[Mm]onthly",Token.category==JJ}|{Token.string==~"[Pp]resent",Token.category==JJ}|{To
ken.string==~"[Ff]urther",Token.category==JJ}|{Token.string==~"[Aa]nnual",Token.category==JJ}
|{Token.string==~"[Tt]emporal",Token.category==JJ}|{Token.string==~"[Cc]ontemporary",Token.category==J
J}|{Token.string==~"[Ee]lderly",Token.category==JJ}|{Token.string==~"[Uu]ltimate",Token.category==JJ}
|{Token.string==~"[Tt]imely",Token.category==JJ}|{Token.string==~"[Bb]aby",Token.category==JJ}|{Token.
string==~"[Bb]abyish",Token.category==JJ}|{Token.string==~"[Tt]eenage",Token.category==JJ}
|{Token.string==~"[Aa]ntique",Token.category==JJ}|{Token.string==~"[Oo]ld-
fashioned",Token.category==JJ}|{Token.string==~"[Yy]outhful",Token.category==JJ}|{Token.string==~"[Mm]
ature",Token.category==JJ}
|{Token.string==~"[Aa]dolescent",Token.category==JJ}|{Token.string==~"[Ii]nfantile",Token.category==JJ
}|{Token.string==~"[Bb]ygone",Token.category==JJ}|{Token.string==~"[Rr]ecent",Token.category==JJ}
|{Token.string==~"[Ee]arly",Token.category==JJ}|{Token.string==~"[Ll]ate",Token.category==JJ}|{Token.s
tring==~"[Ll]ast",Token.category==JJ}
|{Token.string==~"[Pp]rehistoric",Token.category==JJ}
)

Rule: Time
Priority: 35
(
(Atrib_time):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Time");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_time");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test26
Input:  Token Lookup NPJK
Options: control = appelt

Macro:Atrib_touch
(
{Token.string==~"[Mm]elted",Token.category==JJ}|{Token.string==~"[Pp]rickly",Token.category==JJ}|{Toke
n.string==~"[Uu]neven",Token.category==JJ}|{Token.string==~"[Ss]oft",Token.category==JJ}
|{Token.string==~"[Ss]ilky",Token.category==JJ}|{Token.string==~"[Vv]elvety",Token.category==JJ}|{Toke
n.string==~"[Bb]umpy",Token.category==JJ}|{Token.string==~"[Ss]mooth",Token.category==JJ}
|{Token.string==~"[Gg]rainy",Token.category==JJ}|{Token.string==~"[Cc]oarse",Token.category==JJ}|{Toke
n.string==~"[Pp]itted",Token.category==JJ}
|{Token.string==~"[Ss]caly",Token.category==JJ}|{Token.string==~"[Pp]olished",Token.category==JJ}|{Tok
en.string==~"[Gg]lossy",Token.category==JJ}|{Token.string==~"[Ll]umpy",Token.category==JJ}
|{Token.string==~"[Ww]iry",Token.category==JJ}|{Token.string==~"[Ss]cratchy",Token.category==JJ}|{Toke
n.string==~"[Gg]lassy",Token.category==JJ}
|{Token.string==~"[Bb]oling",Token.category==JJ}|{Token.string==~"[Bb]reeze",Token.category==JJ}|{Toke
n.string==~"[Cc]reepy",Token.category==JJ}
|{Token.string==~"[Cc]uddly",Token.category==JJ}|{Token.string==~"[Cc]urly",Token.category==JJ}|{Token
.string==~"[Dd]amaged",Token.category==JJ}
|{Token.string==~"[Dd]usty",Token.category==JJ}|{Token.string==~"[Ff]laky",Token.category==JJ}|{Token.
string==~"[Ff]luffy",Token.category==JJ}

)

Rule: Touch
Priority: 35
(
(Atrib_touch):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Touch");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_touch");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test20
Input:  Token Lookup NPJK
Options: control = appelt

Macro:Atrib_type
(
{Token.string==~"[Ss]helf-type",Token.category==JJ}|{Token.string==~"[Pp]ort-
type",Token.category==JJ}|{Token.string==~"[Tt]op",Token.category==JJ}
|{Token.string==~"[Aa]rtistic",Token.category==JJ}|{Token.string==~"[Dd]own",Token.category==JJ}|{Toke
n.string==~"[Ff]ederal",Token.category==JJ}|
{Token.string==~"[Dd]ramatic",Token.category==JJ}|{Token.string==~"[Cc]arry-on",Token.category==JJ}|
{Token.string==~"[Rr]isk-based",Token.category==JJ}|{Token.string==~"[Ii]n-
flight",Token.category==JJ}|
{Token.string==~"[Oo]n-
board",Token.category==JJ}|{Token.string==~"[Cc]ohesive",Token.category==JJ}|{Token.string==~"[Ii]ntel
ligence-based",Token.category==JJ}|
{Token.string==~"[Ii]n-depth",Token.category==JJ}|{Token.string==~"[Ii]ndustry-
wide",Token.category==JJ}|{Token.string==~"[Aa]viation-related",Token.category==JJ}|
{Token.string==~"[Rr]apid-deployment",Token.category==JJ}|{Token.string==~"[Cc]ost-
effective",Token.category==JJ}|{Token.string==~"[Tt]op-notch",Token.category==JJ}
|{Token.string==~"[Ss]exual",Token.category==JJ}
)

Rule: Type
Priority: 35
(
(Atrib_type):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Type");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_type");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test15
Input:  Token Lookup NPJK
Options: control = appelt

/
**************************************************************************************************
*************************************************/
Macro:Atrib_use
(
{Token.string==~"[Ee]lectrical",Token.category==JJ}|{Token.string==~"[Pp]olitical",Token.category==JJ}
|{Token.string==~"[Pp]sychological",Token.category==JJ}|{Token.string==~"[Pp]hysical",Token.category==
JJ}
|{Token.string==~"[Hh]istorical",Token.category==JJ}|{Token.string==~"[Ff]inancial",Token.category==JJ
}|{Token.string==~"[Mm]edical",Token.category==JJ}|{Token.string==~"[Tt]echnical",Token.category==JJ}
|{Token.string==~"[Ll]ogical",Token.category==JJ}|{Token.string==~"[Ii]nternal",Token.category==JJ}|{T
oken.string==~"[Ee]xternal",Token.category==JJ}|{Token.string==~"[Mm]echanical",Token.category==JJ}
|{Token.string==~"[Pp]hilosophical",Token.category==JJ}|{Token.string==~"[Ee]nvironmental",Token.categ
ory==JJ}|{Token.string==~"[Ii]ndustrial",Token.category==JJ}|
{Token.string==~"[Aa]gronomic",Token.category==JJ}|{Token.string==~"[Ss]tatistical",Token.category==JJ
}|{Token.string==~"[Ss]easonal",Token.category==JJ}
|{Token.string==~"[Ee]cological",Token.category==JJ}|{Token.string==~"[Ss]tructural",Token.category==J
J}|{Token.string==~"[Ww]rongful",Token.category==JJ}|{Token.string==~"[Cc]riminal",Token.category==JJ}
|{Token.string==~"[Pp]harmaceutical",Token.category==JJ}|{Token.string==~"[Cc]ritical",Token.category=
=JJ}|{Token.string==~"[Oo]perational",Token.category==JJ}|{Token.string==~"[Aa]dditional",Token.catego
ry==JJ}
|{Token.string==~"[Aa]gricultural",Token.category==JJ}|{Token.string==~"[Ee]mpirical",Token.category==
JJ}|{Token.string==~"[Tt]opical",Token.category==JJ}|{Token.string==~"[Uu]niversal",Token.category==JJ
}
|{Token.string==~"[Ee]ducational",Token.category==JJ}|{Token.string==~"[Tt]echnological",Token.categor
y==JJ}|{Token.string==~"[Cc]ollegial",Token.category==JJ}|{Token.string==~"[Oo]fficial",Token.category
==JJ}
|{Token.string==~"[Cc]ultural",Token.category==JJ}|{Token.string==~"[Ll]exical",Token.category==JJ}|{T
oken.string==~"[Cc]entral",Token.category==JJ}
)

Rule: Use
Priority: 35
(
(Atrib_use):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Use");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_use");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test36
Input:  Token Lookup NPJK
Options: control = appelt

Macro:Atrib_value
(
{Token.string==~"[Gg]ood",Token.category==JJ}|{Token.string==~"[Nn]ice",Token.category==JJ}|{Token.str
ing==~"[Gg]reat",Token.category==JJ}
|{Token.string==~"[Aa]lright",Token.category==JJ}|{Token.string==~"[Bb]ad",Token.category==JJ}|{Token.
string==~"[Ii]mport",Token.category==JJ}
|{Token.string==~"[Ll]ovely",Token.category==JJ}|{Token.string==~"[Ff]ine",Token.category==JJ}|{Token.
string==~"[Ff]unny",Token.category==JJ}
|{Token.string==~"[Ii]nteresting",Token.category==JJ}|{Token.string==~"[Oo]kay",Token.category==JJ}
)

Rule: Value
Priority: 35
(
(Atrib_value):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Value");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_value");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test9
Input:  Token Lookup NPJK
Options: control = appelt

/
**************************************************************************************************************
*************************************************/
Macro:Atrib_weather
(
{Token.string==~"[Rr]ainy",Token.category==JJ}|{Token.string==~"[Ss]tormy",Token.category==JJ}|{Token.
string==~"[Ss]unny",Token.category==JJ}|{Token.string==~"[Ww]indy",Token.category==JJ}
|{Token.string==~"[Ss]nowy",Token.category==JJ}|{Token.string==~"[Dd]amp",Token.category==JJ}|{Token.s
tring==~"[Dd]ry",Token.category==JJ}
|{Token.string==~"[Ff]oggy",Token.category==JJ}|{Token.string==~"[Oo]vercast",Token.category==JJ}|{Tok
en.string==~"[Cc]loudy",Token.category==JJ}
|{Token.string==~"[Mm]ild",Token.category==JJ}|{Token.string==~"[Aa]tmospheric",Token.category==JJ}|{T
oken.string==~"[Ww]ind-lashed",Token.category==JJ}
)

Rule: Weather
Priority: 35
(
(Atrib_weather):sust2 /*VALORINSTANCIATRIBUTO*/
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE*/
):inst
-->
{
try{
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Weather");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("attributeName", "has_weather");
}
outputAS.add(as.firstNode(),as.lastNode(),"Instancias",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

D.Anexo: Reglas JAPE para extracción de relaciones

```
Phase:Test1
Input:  Token Lookup NPJK
Options: control = appelt

Rule: Relaciones1
Priority: 25
(
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE DOMINIO*/
{Token.string=="and"}
{Token.string=="other"}
({Token.length>1,NPJK.kind=="NPJK"}):sust2 /*CLASE RANGO*/
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Relaciones1");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("relationName", "and_other");
}
outputAS.add(as.firstNode(),as.lastNode(),"Relaciones",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test1
Input:  Token Lookup NPJK
Options: control = appelt

//Regla IS_A o IS_AN
Rule: Relaciones2
Priority: 25
(
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE DOMINIO*/
{Token.string=="is"}
({Token.string=="a"}|{Token.string=="an"}):auxi
({Token.length>1,NPJK.kind=="NPJK"}):sust2 /*CLASE RANGO*/
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet aux =(gate.AnnotationSet)bindings.get("auxi");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

for(Annotation nino2:aux){
String tt =
doc.getContent().getContent(nino2.getStartNode().getOffset(),nino2.getEndNode().getOffset()).toString(
);
features.put("relationName","is_"+tt);
}


/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Relaciones2");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(),as.lastNode(),"Relaciones",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test1
Input:  Token Lookup NPJK
Options: control = appelt

Rule: Relaciones3
Priority: 25
(
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE DOMINIO*/
{Token.string=="is"}
{Token.string=="located"}
{Token.string=="in"}
({Token.length>1,NPJK.kind=="NPJK"}):sust2 /*CLASE RANGO*/
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Relaciones3");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("relationName", "is_located_in");
}
outputAS.add(as.firstNode(),as.lastNode(),"Relaciones",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test1
Input:  Token Lookup NPJK
Options: control = appelt

Rule: Relaciones5
Priority: 25
(
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE DOMINIO*/
{Token.string=="derives"}
{Token.string=="from"}
({Token.length>1,NPJK.kind=="NPJK"}):sust2 /*CLASE RANGO*/
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Relaciones5");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("relationName", "derives_from");
}
outputAS.add(as.firstNode(),as.lastNode(),"Relaciones",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test1
Input:  Token Lookup NPJK
Options: control = appelt

Rule: Relaciones6
Priority: 25
(
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE DOMINIO*/
{Token.string=="contained"}
{Token.string=="in"}
({Token.length>1,NPJK.kind=="NPJK"}):sust2 /*CLASE RANGO*/
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Relaciones6");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("relationName", "contained_in");
}
outputAS.add(as.firstNode(),as.lastNode(),"Relaciones",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test1
Input:  Token Lookup NPJK
Options: control = appelt

Rule: Relaciones7
Priority: 25
(
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE DOMINIO*/
{Token.string=="part"}
{Token.string=="whole"}
({Token.length>1,NPJK.kind=="NPJK"}):sust2 /*CLASE RANGO*/
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Relaciones7");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("relationName", "part_whole");
}
outputAS.add(as.firstNode(),as.lastNode(),"Relaciones",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test1
Input:  Token Lookup NPJK
Options: control = appelt

Rule: Relaciones8
Priority: 25
(
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE DOMINIO*/
{Token.string=="preceded"}
{Token.string=="by"}
({Token.length>1,NPJK.kind=="NPJK"}):sust2 /*CLASE RANGO*/
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Relaciones8");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
features.put("relationName", "preceded_by");
}
outputAS.add(as.firstNode(),as.lastNode(),"Relaciones",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

```
Phase:Test1
Input:  Token Lookup NPJK
Options: control = appelt

//Regla IS_A o IS_AN
Rule: Relaciones9
Priority: 25
(
({Token.length>1,NPJK.kind=="NPJK"}):sust1 /*CLASE DOMINIO*/
{Token.string=="is"}
({Token.string=="a"}|{Token.string=="an"}):auxi
({Token.length>1,NPJK.kind=="NPJK"})
{Token.category==CC}
({Token.length>1,NPJK.kind=="NPJK"}):sust2 /*CLASE RANGO*/
):inst
-->
{
try {
AnnotationSet as =(gate.AnnotationSet)bindings.get("sust2");
AnnotationSet kk =(gate.AnnotationSet)bindings.get("sust1");
AnnotationSet aux =(gate.AnnotationSet)bindings.get("auxi");

FeatureMap features =Factory.newFeatureMap();

/* EXTRAER EL NOMBRE DE LA CLASE*/
for(Annotation nino:kk){
String clase =
doc.getContent().getContent(nino.getStartNode().getOffset(),nino.getEndNode().getOffset()).toString();
features.put("className",clase);
}

for(Annotation nino2:aux){
String tt =
doc.getContent().getContent(nino2.getStartNode().getOffset(),nino2.getEndNode().getOffset()).toString(
);
features.put("relationName","is_"+tt);
}


/*EXTRAER EL NOMBRE DE LA INSTANCIA*/
for(Annotation ann:as){
String instanceName =
doc.getContent().getContent(ann.getStartNode().getOffset(),ann.getEndNode().getOffset()).toString();
features.put("kind",ann.getFeatures().get("kind"));
features.put("rule","Relaciones9");
features.put("representationId",ann.getId());
features.put("id",ann.getId());
features.put("corefchain", null);
features.put("instanceName", instanceName);
}
outputAS.add(as.firstNode(),as.lastNode(),"Relaciones",features);

}
catch(Exception e)
{
e.printStackTrace();
}
}
```

E. Anexo: Documentos de prueba

Documento Dominio Showbiz

The O. J. Simpson murder case (officially the People of the State of California v. Orenthal James Simpson) was a criminal trial held at the Los Angeles County Superior Court in California. The trial spanned from the jury's swearing-in on November 2, 1994,[1] to opening statements on January 24, 1995,[2] to a verdict on October 3, 1995.[3] The former professional football star and actor O. J. Simpson was tried on two counts of murder after the deaths of his ex-wife, Nicole Brown Simpson, and waiter Ronald Lyle Goldman, in June 1994. The case has been described as the most publicized criminal trial in American history.[4] Simpson was acquitted after a trial that lasted more than eight months.[5]

Simpson hired a high-profile defense team, initially led by Robert Shapiro[6][7][8] and subsequently led by Johnnie Cochran, and which also included F. Lee Bailey, Alan Dershowitz, Robert Kardashian, Gerald Uelmen, Robert Blasier, and Carl E. Douglas, with two more attorneys specializing in DNA evidence: Barry Scheck and Peter Neufeld. Los Angeles County believed it had a solid prosecution case, but Cochran was able to persuade the jurors that there was reasonable doubt about the DNA evidence (a relatively new form of evidence in trials at the time)[9] – including that the blood-sample evidence had allegedly been mishandled by lab scientists and technicians – and about the circumstances surrounding other exhibits.[10] Cochran and the defense team also alleged other misconduct by the Los Angeles Police Department. Simpson's celebrity and the lengthy televised trial riveted national attention on the so-called "Trial of the Century". By the end of the criminal trial, national surveys showed dramatic differences in the assessment of Simpson's guilt between most black and white Americans.[11]

Later, both the Brown and Goldman families sued Simpson for damages in a civil trial that came to a total of $40 million. On February 6, 1997, a jury unanimously found there was a preponderance of evidence to hold Simpson liable for damages in the wrongful death of Goldman and battery of Brown.[12] On February 21, 2008, a Los Angeles court upheld a renewal of the civil judgment against him.[13]

Documento Dominio Freeze Drying

Lyophilization or freeze drying is the method of drying any pharmaceutical or food products
in the frozen state. The product in solution form is first frozen to ice by applying very low temperature and then removing this ice from the pharmaceutical products by applying low temperature drying in low pressure. Freeze drying works on the principle of sublimation in which liquid state is converted in the solid state below the triple point of water.

For freeze drying? [1]
1. To stabilize highly degradable protein drugs.
2. To stabilize heat sensitive and chemically unstable solutions.
3. Low particulate contamination.
4. Low temperature drying process.
5. Compatible with aseptic/sterile processing.
6. To get amorphous form of drug desirable for solubility in freeze drying.

Desirable freeze drying characterstics [4]
1. Intact cake of products.
2. Sufficient strength in terms of assay pH.
3. Uniform color of products.
4. Sufficient drying of products.
5. Sufficient porosity of finally dried products.
6. Chemical stability of products.

Advantages
1. Heat labile costly drug product can be stabilized in dry form.
2. Increases shelf life of product during storage condition.

Disadvantages of lyophilization Technology
1. Very costly technology increases the cost of drug product.
2. Require monitoring of each and every step of lyophilization and time consuming process.

Documento Dominio Mental imagery

Mental imagery and self-talk strategies are implemented by athletes in order to regulate arousal, reduce maladaptive behaviors, reconstruct negative thoughts, and to increase one's concentration and focus. DeFrancesco and Burke (1997) reported that imagery techniques were found to be the most common strategies employed by both female and male professional tennis players. Lejuene, Decker, and Sanchez (1994) studied the training styles of 40 novice table tennis players and found that "imagining oneself successfully completing a sports skill in the absence of the actual movement or activity increases the probability of improving one's performance" (p. 627). In addition, Mckenzie and Howe (1997) reported that engaging in a 15-week imagery training program improved accuracy scores among dart throwers when compared to participants not exposed to any imagery training. Peluso (2000) reported that participants who engaged in relevant imagery practice increased performance on both a mirror tracing and jack catching task when compared to participants in non-relevant, relaxation, and control conditions. It is widely accepted that that imagery is a powerful and important psychological tool in the enhancement of athletic performance. Within an appropriate training program imagery skills can increase self-awareness, facilitate skill acquisition and maintenance, build self-confidence, control emotions relieve pain, regulate arousal and enhance preparation. The use of imagery is most effective for tasks that have a high "cognitive" component and advanced athletes benefit more from it than beginners also it is seen to be most effective if it precedes physical practice. There is a wide range of imagery abilities between athletes even at elite level and its implementation, so for this reason the starting point of this intervention must be training in effective imagery techniques. The more imagery is done the better the individual will become, it is important to gauge what level of imaging skills the individual has at the outset. This may include issues of lack of concentration, embarrassment or even an inability to translate a physical skill, at which they are highly proficient, into effective imagery. Athletes who image the arousal stress and anxiety that may accompany performance and the use of imagery which includes being in control in difficult situations or individuals that image themselves being mentally tough have higher levels of self-confidence. As self-confidence is recognized by most athletes and coaches as an important cognitive determinant of athletic performance the use of this type of imagery is to be encouraged. However the use of this imagery can be counterproductive for individuals who do not deal with cognitive state anxiety well as they will be unable to feel positive and image a believable positive outcome. For many years the use of cognitive imagery, athletes imaging themselves performing skills, just before competition was seen also as a way to increase self-confidence, however later studies refuted this. There was no correlation found between cognitive imagery and self-confidence and its use is purely one of preparing the motor system for action. Previous experience is the strongest predictor cognitive sate anxiety, competitors who have previously succeeded report lower levels than those that have not. The ability to utilize experienced images rather than predictive images gives a much stronger effect. Thus, the present research tries to consider the mental imagery effect upon the reduction of athletes' anxiety during sport performance.

Documento Dominio Agriculture (Extensión 8 páginas)

Corn Grain Yield Response to Crop Rotation and Nitrogen over 35 Years

Trenton F. Stanger* and Joseph G. Lauer

Corn

ABSTRACT

Crop rotation and N are management methods that can increase corn (*Zea mays* L.) grain yields. Our objective was to determine the corn grain yield response to six crop rotation sequences and four N rates in a long-term (35-yr) study. The rotations were continuous corn (CC), corn–alfalfa (*Medicago sativa* L.) (CA), corn–soybean [*Glycine max* (L.) Merr.] (CS), corn-corn-corn-alfalfa-alfalfa (CCCAA), corn-corn-oat (*Avena sativa* L.) with alfalfa seeding-alfalfa-alfalfa (CCOaAA), and corn-soybean-corn-oat with alfalfa seeding–alfalfa (CSCOaA). From 1970 to 2004, first-yr corn grain yields (*C*CCAA, *C*COaAA, and *C*SCOaA) increased from 79 to 100 kg ha^{-1} yr^{-1}. Increasing N rates did not influence grain yield trends, indicating that an alfalfa crop produced the N required by first-yr corn. However, 224 kg N ha^{-1} was needed to improve second and third-yr grain yield trends 69 and 58 kg ha^{-1} yr^{-1}, respectively. Grain yield trends for CC did not improve despite increasing N treatments, although grain yield tended to increase over time at 224 kg N ha^{-1} ($P < 0.10$). From 1989 to 2004, corn grain yield trends of CA and CS decreased by 161 kg ha^{-1} yr^{-1} if no N was added. The 2-yr rotation was not sufficient to improve grain yield trends, whereas the 5-yr rotation was able to enhance corn grain yield and decrease the need for fertilizer N. Effects on pathogens and insects were not evaluated but warrant further investigations. Overall, this data shows that extended rotations involving forage crops reduce N inputs, increase corn grain yields, and are more agronomically sustainable than current short-term rotations.

Agricultural productivity gains since the 1950s resulted from the development of farming systems that rely extensively on external inputs of energy and chemicals to replace management and on-farm resources (Oberle, 1994). Cropping sequence in the midwestern United States has increasingly evolved to a greater reliance on the CS rotation and even CC over the last half of the century. There are many reasons for this, among them the development of effective fertilizers and pesticides, government policies, and favorable economics (Porter et al., 2003).

Nitrogen has been considered one of the best crop-input investments that a farmer can make in terms of return on dollars spent (Pikul et al., 2005); however, N is the most expensive nutrient for growing grain crops. Bundy et al. (1999) estimated that in the 12-states of the north central United States, at least 3.6 Tg of N fertilizer were applied annually to corn at a cost of $800 million.

Many crops respond dramatically to applications of N fertilizer. Nitrogen fertilizer is universally accepted as a key input to high corn grain yield and optimum economic return. In the Midwest, the primary philosophical approach to developing an N fertilizer recommendation for corn is to consider, as independent variables, yield goal, economic return, management level, and some measure of the inherent differences in soil productivity (Oberle and Keeney, 1990). Overapplication is more frequent since producers have an economic incentive to err more frequently in that direction. From a crop production perspective, the cost of unneeded N fertilizer in areas of overapplication is less than the cost of lost yield potential in areas of underapplication (Scharf et al., 2005). This is not true from an environmental perspective (Hallberg, 1989; Yadav, 1997).

According to Heichel (1978) and Pimentel et al. (1978), continuous rotation of corn and soybean cannot be sustained without substantial additions of fertilizers. Recent statistics (USDA-National Agricultural Statistics Service, 2006a) show for example that corn grown in the United States receives around 4.5 Tg of N annually. These high application levels result in low N use efficiency (NUE), currently estimated at only 33% (Raun and Johnson, 1999) for world cereal grain production systems. In addition to economic losses, N overapplication results in environmental contamination through nitrate N runoff or leaching, making nitrate N the most common contaminant found in the surface and ground water of the Corn Belt (Council for Agricultural Science and Technology, 1999).

Increased diversity of crops grown in rotation enhances sustainability of agriculture systems because crops grown in rotation, with similar off-farm inputs, have greater yield than those grown in monoculture (Mannering and Griffith, 1981; Dick et al., 1986; Higgs et al., 1990). Other key benefits of including a forage or pasture crop consist of improved soil quality (i.e., increased C retention in the surface horizon) and a more

T.F. Stanger, Monsanto Company, 1920 Fifth St., Davis, CA 95616; J.G. Lauer, Dep. of Agronomy, 1575 Linden Dr., Univ. of Wisconsin, Madison, WI 53706. Funded by CSREES project WIS0 142-4897. Received 20 Aug. 2007. *Corresponding author (trenton.floyd.stanger@monsanto.com).

Published in Agron. J. 100:643–650 (2008).
doi:10.2134/agronj2007.0280

Abbreviations: CA, corn-alfalfa; CC, continuous corn; CCCAA, corn-corn-corn-alfalfa-alfalfa; CCOaAA, corn–corn-oat with alfalfa seeding-alfalfa-alfalfa; CS, corn-soybean; CSCOaA, corn-soybean-corn-oat with alfalfa seeding-alfalfa; NUE, nitrogen use efficiency. *Italics in a rotation code indicate the year of the rotation under discussion.*

Table 1. Crop rotations and N rates at Lancaster, WI, used to evaluate the influence of crop rotation and N on the rotation effect of corn.†

Crop rotation treatments			N Treatments	
1966–1976	1977–1986	1987–2004	1967–1976	1977–2004
			—kg N ha^{-1}—	
CC‡	**CC**	**CC**	**0**	**0**
CSCOaA	**CSCOaA**	**CSCOaA**	**84**	**56**
CCCOaA	**CCCAA**	**CCCAA**	**168**	**112**
CCOaAA	**CCOaAA**	**CCOaAA**	**336**	**224**
COaAAA	CCAA	**CA**		
	AA	**CS**		
		AA		

† C, corn; S, soybean; Oa, oat with alfalfa seeding; A, alfalfa.

‡ The rotations in bold were included in the analysis.

even distribution of labor needs and risk due to climate or market conditions than those involving only grain or fiber crops (Magdoff and van Es, 2000).

Crop rotations that include legumes also increase soil N levels (Peterson and Varvel, 1989; Raimbault and Vyn, 1991). Crop rotation has also been shown to improve NUE by reducing requirements for external input of fertilizer N. Bruulsema and Christie (1987) found that a single-year of alfalfa or red clover (*Trifolium pretense* L.) resulted in corn yields (8.8 and 8.5 Mg ha^{-1}, respectively) equivalent to that obtained from applying 90 to 125 kg ha^{-1} of fertilizer N. Fox and Piekielek (1988) extended the evaluation period to 3 yr of alfalfa managed as hay and reported that there was no significant grain yield response (10.3 Mg ha^{-1}) to fertilizer N for first-year corn.

Despite the benefits of crop rotations that include legumes, the infrastructure developed and devoted to corn and soybean has resulted in a 500% increase in harvested area of these crops in the United States between 1950 and 2003 (USDA-National Agricultural Statistics Service, 2004). During that same period, oat production declined 90%, and although hay production increased because of better yields, the land area devoted to it decreased more than 15%. This occurred for several reasons including simplicity and similar equipment requirements as farm size increased, commodity programs that emphasized short-term profit, public and private research and development efforts devoted to genetic improvement of corn and soybean, and increased food and industrial uses for both corn and soybean oils and various by-products (Karlen, 2004). Expansion of the simplified CS system has tremendous economic and world trade benefits because of the many products and materials developed from those crops (Karlen et al., 2006).

In 1966, a multiple crop rotation experiment was established to evaluate the profitability and agronomic sustainability of corn-based crop rotations. Forty years later, the experiment has become one of the longest running rotation studies in the United States. Previous papers published from this experiment have looked at the effect of legumes on subsequent corn crops (Baldock et al., 1981), corn N recommendations based on yield goals vs. soil specific data (Vanotti and Bundy, 1994a, 1994b), the frequency of N fertilizer carryover (Vanotti and Bundy, 1994c), soybean effects on soil N availability (Vanotti and Bundy, 1995), and the effect of extended rotations improving soil quality and profitability (Karlen et al., 2006). The objective of this paper was to determine the effect of crop rotation and applied N on corn grain yield trends for a 35-yr period.

MATERIALS AND METHODS

Data from a long-term crop rotation study located in southwestern Wisconsin [University of Wisconsin Agricultural Research Station-Lancaster (42°50′ N, 90°47′ W; elevation 324 m above mean sea level)] near Lancaster was used for this study. A randomized complete block in a split-plot design with two replications of 21 treatments was established to test the rotation effect by having each crop phase of every rotation represented each year. To accommodate all possible crop phases of the rotations and four fertilizer treatments, 168 plots (6.1 by 9.1 m) were established in 1966. Thus, for CC, there was one plot within each replication, and for CS there was one corn plot and one soybean plot within each replication. Rotation treatments have changed over time (Table 1). The crop sequence plots were split to accommodate four N rate treatments. From 1967 to 1976, N rates were 0, 84, 168, and 336 kg N ha^{-1}, but since 1977, the annual rates have been 0, 56, 112, and 224 kg N ha^{-1} for corn only (Table 1). Nitrogen fertilizer treatments were broadcast by hand each spring as ammonium nitrate (NH_4NO_3).

Tillage has varied over time. Corn following corn and oat and alfalfa seedbed preparation has always been fall chisel plowed followed by spring disking and cultimulching before planting. Corn following soybean has been no-till since 1994, while corn following alfalfa and soybean following corn have been no-till since 1999. Soil fertility samples were collected and analyzed every 3 yr, and uniform rates of P and K fertilizers were applied as needed to maintain optimum to high soil-test levels. Herbicides and cultivation were used for weed control as needed. Cultivars varied over time but were always improved selections developed for the region. The alfalfa, which was seeded with oat, has not been harvested during the seeding year following oat harvest. For alfalfa that was independently established, two harvests were taken during the seeding year, except for rotations with 1-yr alfalfa, where the alfalfa was killed during the fall of the same year following a third cutting by plowing (before 1999) or through the use of appropriate herbicides (2000 onward). For rotations with 2 or 3 yr of alfalfa following establishment, three harvests were taken.

We chose to focus only on corn yields using a subset of crop rotations to test our objective regarding crop rotation. The rotations of interest were CC, CS, CA, CSCOaA, CCCAA, and CCOaAA. Italics in a rotation code indicate the year of corn in the rotation under discussion. We treated CCCOaA and CCCAA in this analysis as one continuing rotation (see Table 1). The only difference was that in CCCAA, oats were not used as a nurse crop and the alfalfa was independently established.

The Lancaster cropping systems study is comprised of multiple crop rotations that take varying amounts of time to complete a rotation sequence. For example, CC takes 1 yr, CS takes 2 yr, and CSCOaA takes 5 yr (Table 1). Previous rotation experiments have always analyzed the data on a year by year basis. However, the traditional analysis using years can be expanded to analyze both spatial and temporal trends based on the average grain yields produced in the period it took to complete the cycle. By doing this, we can see how the rotations per-

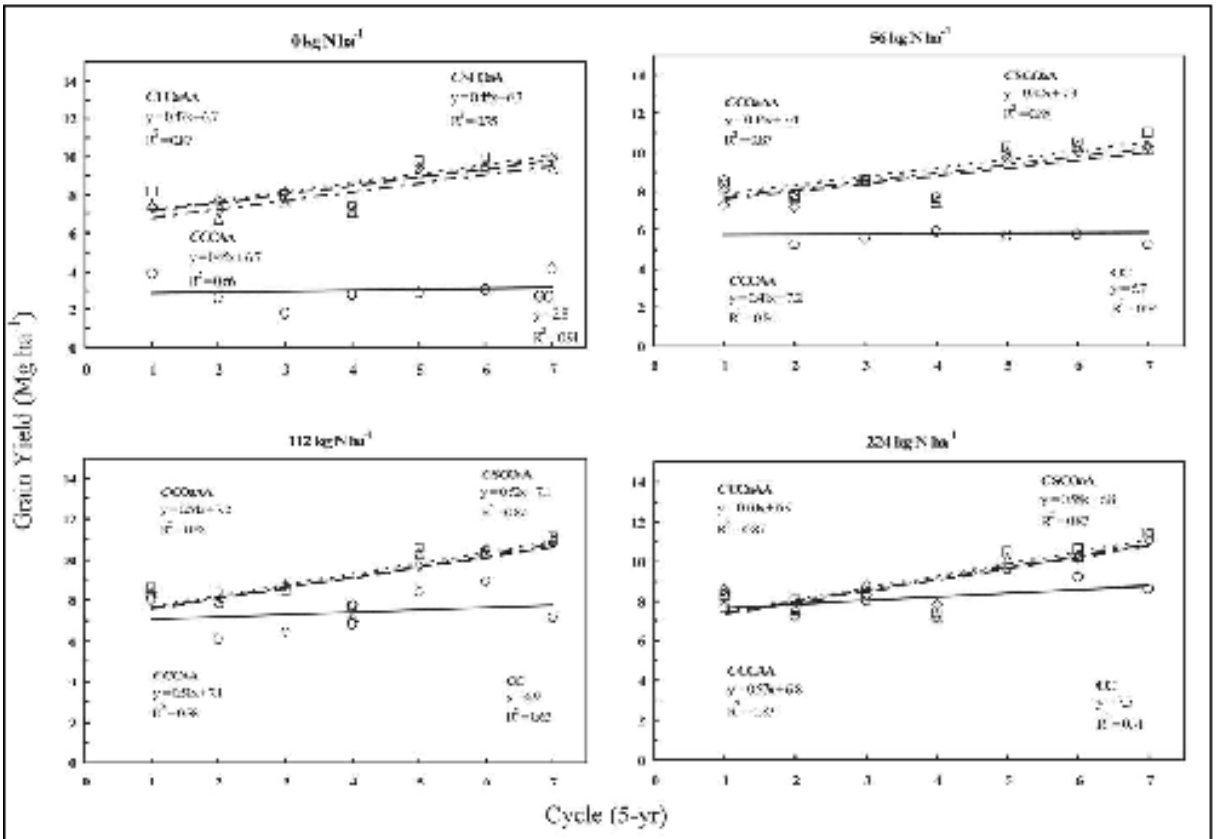


Fig. 1. Relationship between grain yield and time for each N rate and rotation sequence for the first year of corn at Lancaster, WI from 1970 to 2004. C, corn; S, soybean; Oa, oat with alfalfa seeding; A, alfalfa. CC, circle, ————; CCCAA, diamond, ——·——; CCOaAA, square, – – –; CSCOaA, triangle, ——– – ——. Cycles: 1, 1970–1974; 2, 1975–1979; 3, 1980–1984; 4, 1985–1989; 5, 1990–1994; 6, 1995–1999; 7, 2000–2004. Before 1977, N rates were 0, 84, 168, and 336 kg N ha^{-1}. Italic indicates year of rotation for the corn crop being evaluated.

formed when they returned to the same plot of land allowing data analysis across both time and space. Hence, we analyzed the data in groups of either 2- or 5-yr depending on the length of the rotation cycle using CC as our control.

Data were analyzed by covariate analysis using the PROC MIXED procedure (Littell et al., 1996) of SAS (SAS Institute, 2002). The covariate was rotation cycle using the mean grain yield from across the years for that cycle. Repeated measures analysis (SAS Institute, 2002) with the first-order autoregressive variance structure was used to evaluate time and space effects on grain yield. Regression slopes of each year of corn within each rotation sequence were evaluated to determine the long-term effects of various crop rotations and different N fertilization rates on grain yield. Each regression slope was compared to a slope of zero to determine if over time the rotation treatments were improving or deteriorating, and to each other to determine if the relative slopes of each treatment are common or not. For determining the expected mean squares, appropriate F-tests in the analysis of covariance and t tests in the analysis of slopes-equal-to-zero and common-slopes, random effects were rep(year) and year. Least square means of the fixed effects were computed, and the PDIFF option of the LSMEANS statement was used to display the differences among least square means for comparison. This option uses Fisher's protected least significant difference, and comparisons were conducted at $P \leq 0.05$. The P value was corrected for the common-slope contrasts using the Bonferroni procedure for multiple comparisons (George et al., 2005). The final model used to determine each of the regression equations was attained using backward stepwise selection. This procedure starts with the full model and sequentially deletes factors and their interactions. The factor producing the smallest F value is deleted at each stage and the model is complete when the factors remaining in the model have a $P \leq 0.05$. Intensity (kg ha^{-1} yr^{-1}) of significant yield treads was calculated from the final regression equations by calculating the difference in average yields between the first and last cycle, then dividing by the total number of years observed. The R^2 was derived using the predicted values calculated by PROC MIXED $\{R^2 = 1 - [(y_{ij} - \hat{y}_{\text{Pred}})^2/(y_{ij} - y_{\text{grand mean}})^2]\}$.

RESULTS AND DISCUSSION

Five-Year Rotations–First-Year Corn (1970–2004)

First-yr corn grain yields (*CCCAA*, *CCOaAA*, and *CSCOaA*) increased from 79 to 100 kg ha^{-1} yr^{-1} with increasing N rates (0 and 224 kg N ha^{-1}, respectively) (Fig. 1). Grain yield trends for CC did not improve relative to a slope of zero despite increasing N treatments; although grain yield tended to increase over time at 224 kg N ha^{-1} ($P < 0.10$). These results imply that either hybrids have not improved since 1970 or that

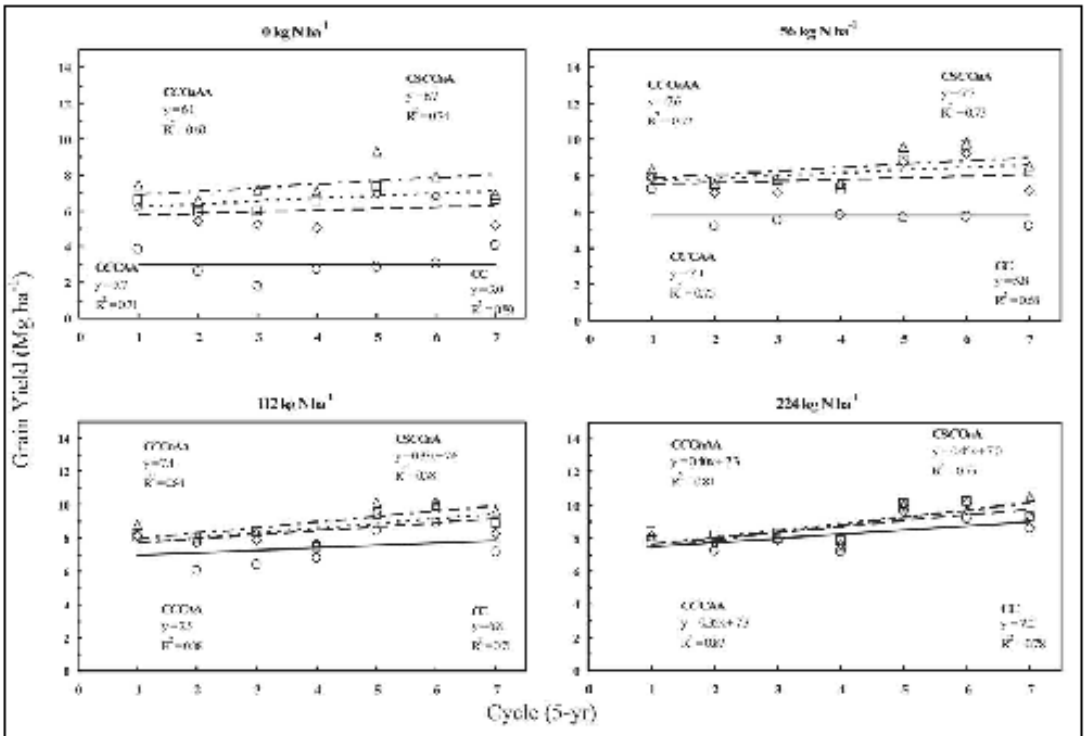


Fig. 2. Relationship between grain yield and time for each N rate and rotation sequence for the second year of corn at Lancaster, WI from 1970 to 2004. C, corn; S, soybean; Oa, oat with alfalfa seeding; A, alfalfa. CC, circle, ———; CCCAA, diamond, — — —; CCOaAA, square, – – –; CSCOaA, triangle, — — —. Cycles: 1, 1970–1974; 2, 1975–1979; 3, 1980–1984; 4, 1985–1989; 5, 1990–1994; 6, 1995–1999; 7, 2000–2004. Before 1977, N rates were 0, 84, 168, and 336 kg N ha^{-1}. Italic indicates year of rotation for the corn crop being evaluated.

improved hybrids have kept CC yield trends from declining over time. According to Tollenaar and Wu (1999) the genetic potential of corn hybrids have improved, however the negative effect of growing CC cancel any yield improvement. Currently, with the rapid turnover of hybrids there is no way to separate the two effects and answer this question.

Rotating corn significantly improved grain yield over time for the first-year of corn when compared to CC (Fig. 1). For the 0, 56, and 112 kg N ha^{-1} treatments, grain yield for *C*CCAA, *C*COaAA, and *CSCOaA* rotations improved 67, 72, and 69 kg ha^{-1} yr^{-1}, respectively. These findings agree with Bolton et al. (1976), Higgs et al. (1976), and Welch (1976) who found corn grown in rotation had higher yields than corn grown in monoculture, even in the presence of N, P, or K fertility levels that were not limiting yields. Corn grown in rotation with a legume receives more N than corn grown continuously with no fertilizer N. However, if N is the only cause of yield differences between rotations, then these differences would be expected to disappear if more than adequate N is applied. It appears that N fertilizers do not substitute for crop rotation (Fig. 1).

There was no difference in slope for the first-year of corn when comparing the rotations involving two, three, and four crops at each N rate (Fig. 1). These results suggest that each rotation sequence in this study is equally effective in breaking the yield depression caused by monoculture. Overall, first-year corn yields for N treatments at 224 kg N ha^{-1} of 5-yr rotations improved by 100 kg ha^{-1} yr^{-1} or 1.4% yr^{-1}, which is similar to the national average (USDA-National Agricultural Statistics Service, 2006b).

Nitrogen fertilizer rate had a significant effect on grain yield slopes of CC (Fig. 1). Although, yield trends for CC by N rate did not improve relative to a slope of zero, the data did show the 112 and 224 kg N ha^{-1} treatments improved grain yield trends by 17 and 28 kg ha^{-1} yr^{-1} when compared to the 56 kg N ha^{-1} treatment, respectively. For the two, three, and four-crop rotation sequences there was no difference in relative grain yield trends as N fertilizer rates increased from 0 to 224 kg N ha^{-1}. First-year alfalfa can supply 134 to 168 kg N ha^{-1} for a subsequent corn crop (Bundy et al., 1990). Because N was never limiting for first-year corn, no differences in grain yield trends with increasing N rates were observed.

Five-Year Rotations–Second-Year Corn (1970–2004)

The benefits of crop rotation in relation to corn grain yield improvement diminished following the first year of corn. Grain yields for the CCCAA, CCOaAA, and CSCOaA rotation at 224 kg N ha^{-1} increased 60, 69, and 77 kg ha^{-1} yr^{-1}, respectively (Fig. 2). The CSCOaA rotation also showed a grain yield improvement of 57 kg ha^{-1} yr^{-1} at 112 kg N ha^{-1}. This improvement in corn grain yield appears to be the result of adding 1 yr of soybean between the first and second year of corn in

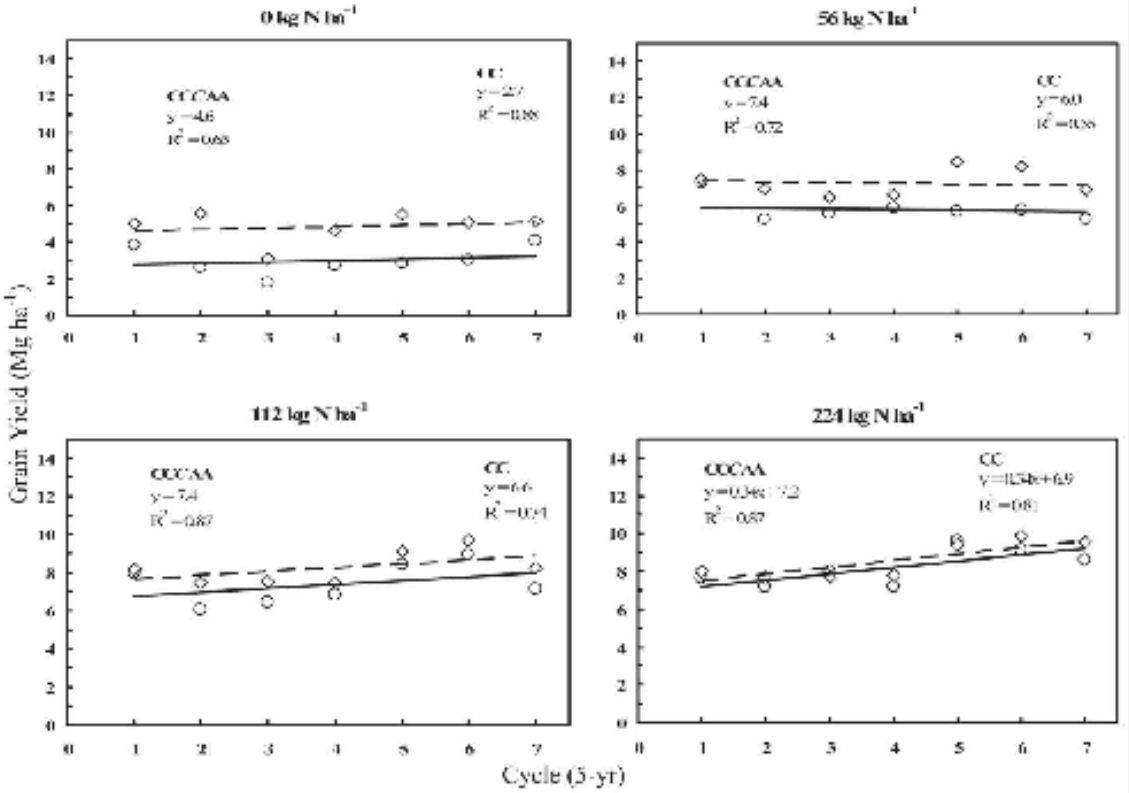


Fig. 3. Relationship between grain yield and time for each N rate and rotation sequence for the third year of corn at Lancaster, WI from 1970 to 2004. C, corn; A, alfalfa. CC, circle, ———; CCCAA, diamond, — — —. Cycles: 1, 1970–1974; 2, 1975–1979; 3, 1980–1984; 4, 1985–1989; 5, 1990–1994; 6, 1995–1999; 7, 2000–2004. Before 1977, N rates were 0, 84, 168, and 336 kg N ha^{-1}. Italic indicates year of rotation for the corn crop being evaluated.

this rotation (Fig. 2). Soybean can supply up to 45 kg N ha^{-1} for a subsequent corn crop (Bundy et al., 1990), explaining the yield improvement at the 112 kg N ha^{-1} rate. Previous research has demonstrated that when corn is grown in rotation with soybean, it yields greater than corn following corn (Baldock et al., 1981; Crookston et al., 1991; Meese et al., 1991; Porter et al., 1997; Pedersen and Lauer, 2002, 2003).

According to the data, the effect of rotating corn for improved corn grain yield over time has diminished when comparing the second year of corn with CC (Fig. 2). Grain yield for CC did not improve, while CCOaAA and CSCOaA improved grain yield by 25 and 33 kg ha^{-1} yr^{-1}, respectively, when compared to CC for the 56 kg N ha^{-1} treatment. No other differences among N treatments for the second year of corn compared to CC were observed. According to Meese et al. (1991) and Pedersen and Lauer (2002), little or no differences were found in grain yield of second-year corn in rotation compared to CC. For the second year of corn in these rotations, the benefits of rotation have largely disappeared.

The N fertilizer rate had a significant effect on grain yield slopes of the second year of corn within their respective rotations (Fig. 2). Similar to the first year of corn, yield trends for CC by N rate did not improve relative to a slope of zero, the data did show the 112 and 224 kg N ha^{-1} treatments improved grain yield trends by 24 and 44 kg ha^{-1} yr^{-1} when compared to the 56 kg N ha^{-1} treatment, respectively. As for the CSCOaA rotation, the 224 kg N ha^{-1} treatment increased grain yield by 44 kg ha^{-1} yr^{-1}, when compared to the 0 kg N ha^{-1} treatments.

Five-Year Rotations–Third-Year Corn (1970–2004)

There was no yield improvement over time for corn from crop rotation by the time the rotation has reached the third year of corn, when compared with CC (Fig. 3). At 224 kg N ha^{-1}, grain yields increased 58 kg ha^{-1} yr^{-1} for both the CC and CCCAA rotation.

The N fertilizer rate had a significant effect on grain yield slopes of the third year of corn within their respective rotations (Fig. 3). Similar to the first and second year of corn, yield trends for CC by N rate did not improve relative to a slope of zero, the data did show the 112 and 224 kg N ha^{-1} treatments improved grain yield trends by 41 and 64 kg ha^{-1} yr^{-1} when compared to the 56 kg N ha^{-1} treatment, respectively. For the CCCAA rotation, the 224 kg N ha^{-1} treatment increased grain yield by 45 kg ha^{-1} yr^{-1}, when compared to the 0 kg N ha^{-1} treatment.

Two-Year Rotations (1989–2004)

There was no yield improvement over time for corn from 2-yr crop rotations, when compared with CC. For the CS rotation at the 0 kg N ha^{-1} treatment, yields actually declined by 189 kg ha^{-1} yr^{-1} since 1989 (Fig. 4). It is unknown as to why

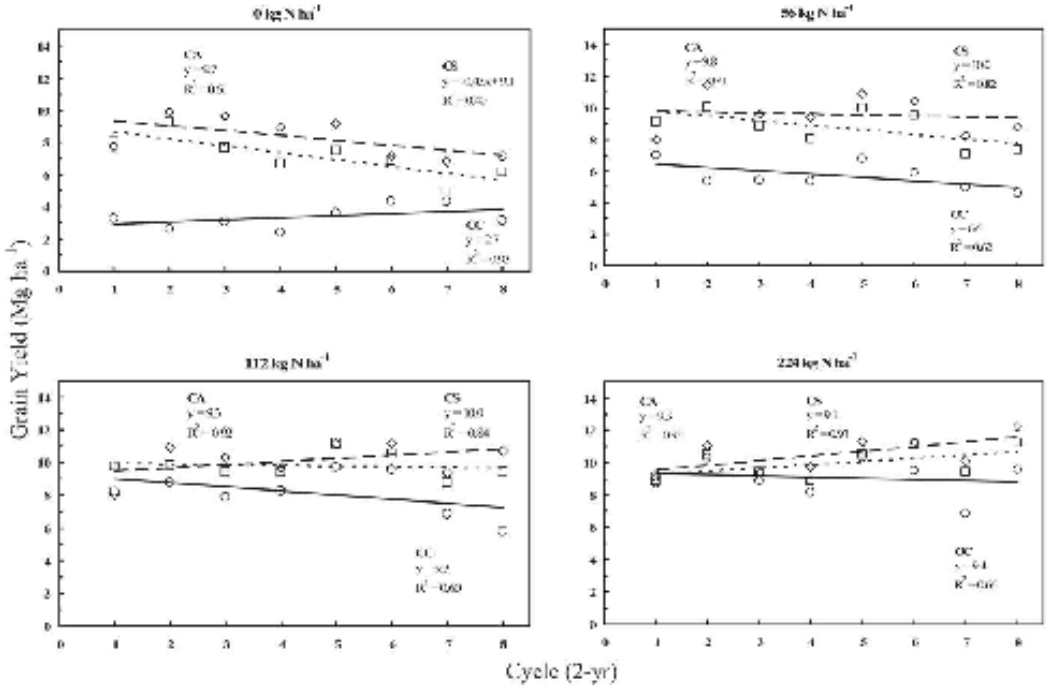


Fig. 4. Relationship between grain yield and time for each N rate and rotation sequence of corn at Lancaster, WI from 1989 to 2004. C, Corn; A, Alfalfa; S, Soybean. CC, circle, ———; CA, diamond, — — —; CS, square, - - - -. Cycles: 1, 1989–90; 2, 1991–1992; 3, 1993–1994; 4, 1995–1996; 5, 1997–1998; 6, 1999–2000; 7, 2001–2002; 8, 2003–2004.

this response was observed. A possible explanation is that corn grain yields in the CS rotation with zero inputs are hampered by lack of adequate N (Porter et al., 2003).

The data suggests the effect of alternating corn with a legume for improved corn grain yield over time appears only when additional N is added to the system (Fig. 4). These 2-yr rotations are not agronomically sustainable, meaning that a single year break between corn crops may not be adequate to eliminate the monoculture yield decline, but only reduce it. The use of N only appears to mask this decline in corn grain yields.

For the 0 kg N ha^{-1} treatment, CC improved grain yield by 249 kg ha^{-1} yr^{-1} when compared to CS (Fig. 4). While alternating corn with soybean at 0 kg N ha^{-1} appears to decrease corn yields over time, adding N fertilizer significantly improved corn grain yield of rotated corn, especially with alfalfa, when compared to CC. For the 112 kg N ha^{-1} treatment, CA improved grain yield by 196 kg ha^{-1} yr^{-1} when compared to CC.

The N fertilizer rate had a significant effect on grain yield slopes for corn that was rotated (Fig. 4). In the CA rotation, the 0 kg N ha^{-1} treatment decreased grain yields over time by 218 and 256 kg ha^{-1} yr^{-1} when compared to the 112 and 224 kg N ha^{-1} treatments, respectively. Likewise, in the CS rotation, the 0 and 56 kg N ha^{-1} treatments decreased grain yields over time by 277 and 211 kg ha^{-1} yr^{-1} when compared to the 224 kg N ha^{-1} treatments, respectively.

These results indicate external inputs of fertilizer mask the true value of crop rotation. According to Porter et al. (2003), one potential way of reducing the amount of external inputs (and associated costs) in a cropping system is to expand the crop rotation into a more diversified crop sequence pattern, thereby taking full advantage of the benefits of crop rotation.

Five-Year vs. Two-Year Rotations (1990–2004)

A comparison was made of both the 5-yr rotations with the 2-yr rotations from 1990 to 2004, on a 5-yr cycle. The slopes of the rotations at each of the N rates were not significantly different from a zero slope, except for the slopes of CA and CS rotations at 0 kg N ha^{-1}, which decreased (Fig. 5). Since 1990, grain yields have declined in the 0 kg N ha^{-1} treatment by 158 and 174 kg ha^{-1} yr^{-1} for the CA and CS rotations, respectively.

In the 0 kg N ha^{-1} treatment, the 2-yr rotations (CA and CS) decreased grain yields over time by 247 and 183 kg ha^{-1} yr^{-1} when compared to CC and the 5-yr rotations, respectively (Fig. 5). For the 56 kg N ha^{-1} treatment, the CS rotation decreased grain yields over time by 159 and 167 kg ha^{-1} yr^{-1} when compared to the *C*CCAA and *C*CO*a*AA rotations, respectively. For the 112 kg N ha^{-1} treatment, the CC rotation decreased grain yields over time by 155 kg ha^{-1} yr^{-1} when compared to the *C*CCAA rotation. Since 1990, in the 224 kg N ha^{-1} treatment, the CC rotation decreased grain yields over time by 160 and 155 kg ha^{-1} yr^{-1} when compared to the *C*CCAA and *C*SCO*a*A rotations, respectively.

Nitrogen fertilizer rate had a significant effect on grain yield slopes for the CA, CC, and CS rotations (Fig. 5). In the CA rotation, the 0 kg N ha^{-1} treatment decreased grain yields over

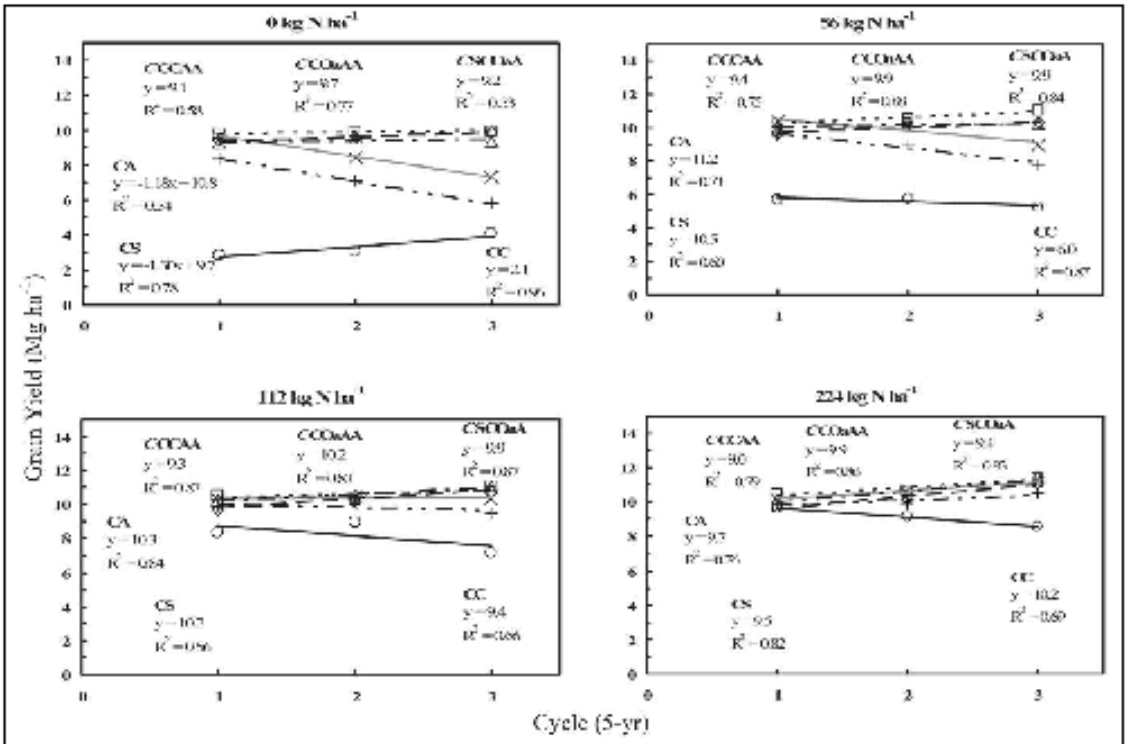


Fig. 5. Relationship between grain yield and time for each N rate and rotation sequence for the first year of corn at Lancaster, WI from 1990 to 2004. C, corn; S, soybean; Oa, oat with alfalfa seeding; A, alfalfa. CC, circle, ———; CCCAA, diamond, — — —; CCOaAA, square, - - - -; CSCOaA, triangle, ——— —; CA, ×, ———; CS, +, ———. Cycles: 1, 1990–1994; 2, 1995–1999; 3, 2000–2004. Italic indicates year of rotation for the corn crop being evaluated.

time by 161 and 220 kg ha^{-1} yr^{-1} when compared to the 112 and 224 kg N ha^{-1} treatments, respectively. Grain yield trends for the 56 kg N ha^{-1} treatment decreased by 157 kg ha^{-1} yr^{-1} when compared to the 224 kg N ha^{-1} treatments. Likewise, in the CS rotation, the 0 kg N ha^{-1} treatment decreased grain yields over time by 197 and 215 kg ha^{-1} yr^{-1} when compared to the 112 and 224 kg N ha^{-1} treatments, respectively, and the 56 kg N ha^{-1} treatment grain yields decreased by 159 kg ha^{-1} yr^{-1} when compared to the 224 kg N ha^{-1} treatment. For the CC rotation, grain yields improved over time for the 0 kg N ha^{-1} treatment by 164 and 151 kg ha^{-1} yr^{-1} when compared to the 112 and 224 kg N ha^{-1} treatments, respectively.

Based on these results, time (2+ yr) along with rotation were required between corn crops to improve corn grain yields. We agree with Randall (2003) and Karlen et al. (2006) that extended rotations involving forage crops may be more agronomically sustainable than current short-term agricultural practices. However, according to Karlen et al. (2006) without the support of federal incentive programs and markets for forage-based products, farmers will hesitate to adopt extended rotations.

CONCLUSION

Extended crop rotations including an alfalfa crop improved first-year corn grain yields with time. First-year corn grain yield trends did not respond to additional N added, demonstrating that an alfalfa crop within an extended crop rotation supplied adequate N. For the second- and third-year of corn only the higher N treatments showed improved corn grain yields over time. The net effect of legumes in improving corn grain yield trends of subsequent corn was not evident for corn that was annually rotated (CA and CS). If no N was added, CA and CS appeared to decrease corn grain yields with time. A single legume crop year was only beneficial in maintaining corn yields over time if N was added to the system. When all rotations were compared (1990–2004), corn grain yields trends of 5-yr crop rotations were significantly better where no N was added and additional N was required for the 2-yr rotations to eliminate this difference. Our data showed a long-term first-year corn grain yield advantage of extended rotations when compared to 2-yr rotations and CC. Nitrogen plays a major role in maintaining and improving corn grain yields in the absence of crop rotation. The addition of N removed the corn grain yield trend differences, along with the passage of time, among crop rotation-phase treatments when CC was compared to the first-year of corn in 5-yr rotations. These results support the argument that extended rotations involving forage crops may be more agronomically sustainable than current short-term (2-yr) rotations, because time (2+ yr) along with rotation and N were required to improve corn grain yields. However, grain yield trends only show the response of long-term management practices on yields and not why this happens. With more than adequate N provided for optimum corn yields, we would expect all yield trends to be similar despite differences in rotational

practices. Further research is needed regarding the influence of above- and below-ground pathogens on corn grain yields and how they may affect yield trends over time.

ACKNOWLEDGMENTS

The authors would like to acknowledge the contributions of Tim Wood for his technical support and providing us with the historical information necessary to complete this paper; Jung Won Mun and Nicholas Keuler for their statistical advising and assistance in data analysis; and to the associate editor and reviewers for the helpful comments. Funding for this research and publication was provided from the USDA Cooperative State Research, Education and Extension Service (CSREES) project WIS0 142-4897.

REFERENCES

Baldock, J.O., R.L. Higgs, W.H. Paulson, J.A. Jackobs, and W.D. Shrader. 1981. Legume and mineral N effects on crop yields in several crop sequences in the Upper Mississippi Valley. Agron. J. 73:885–890.

Bolton, E.F., V.A. Dirks, and J.W. Aylesworth. 1976. Some effects of alfalfa, fertilizer, and lime on corn yield rotation on clay soil during a range of seasonal moisture conditions. Can. J. Soil Sci. 56:21–25.

Bruulsema, T.W., and B.R. Christie. 1987. Nitrogen contribution to succeeding corn from alfalfa and red clover. Agron. J. 79:96–100.

Bundy, L.G., K.A. Kelling, and L. Ward Good. 1990. Using legumes as a nitrogen source. Bull. A3517. Univ. of Wisconsin Coop. Ext. Serv., Madison.

Bundy, L.G., D.T. Walters, and A.E. Olness. 1999. Evaluation of soil nitrate tests for predicting corn nitrogen response in the North Central Region. North Central Reg. Res. Publ. 342. Wisconsin Agric. Exp. Stn., Univ. of Wisconsin, Madison.

Council for Agricultural Science and Technology. 1999. Gulf of Mexico hypoxia: Land and sea interactions. Task Force Rep. 134. CAST, Ames, IA.

Crookston, R.K., J.E. Kurle, P.J. Copeland, J.H. Ford, and W.E. Lueschen. 1991. Rotational cropping sequence affects yield of corn and soybean. Agron. J. 83:108–113.

Dick, W.A., D.M. Van Doren, G.N. Triplett, and J.E. Henry. 1986. Influence of long-term tillage and rotation combinations on crop yields and selected soil parameters. Res. Bull. 1180. Ohio Agric. Res. and Dev. Ctr., The Ohio State Univ., Columbus.

Fox, R.H., and W.P. Piekielek. 1988. Fertilizer N equivalence of alfalfa, birdsfoot trefoil, and red clover for succeeding corn crops. J. Prod. Agric. 1:313–317.

George, E.P., J. Box, S.J. Hunter, and W.G. Hunter. 2005. Statistics for experimenters: Design, innovation, and discovery. 2nd ed. Wiley-Interscience, Hoboken, NJ.

Hallberg, G.R. 1989. Nitrate in groundwater in the United States. p. 35–74. *In* R.F. Follett (ed.) Nitrogen management and groundwater protection. Elsevier, Amsterdam.

Heichel, G.H. 1978. Stabilizing agricultural energy needs: Role of forages, rotation, and nitrogen fixation. J. Soil Water Conserv. 33:279–282.

Higgs, R.L., W.H. Paulsen, J.W. Pendleton, A.F. Peterson, J.A. Jakobs, and W.D. Shrader. 1976. Crop rotations and nitrogen: Crop sequence comparisons on soil of the driftless area of southwestern Wisconsin, 1967–1974. Res. Bull. R2761. Univ. of Wisconsin. College of Agric. and Life Sci., Madison, WI.

Higgs, R.L., A.E. Peterson, and W.H. Paulson. 1990. Crop rotation: Sustainable and profitable. J. Soil Water Conserv. 45:68–70.

Karlen, D.L. 2004. Cropping systems: Rain-fed maize-soybean rotations of North America. p. 358–362. *In* R.M. Goodman (ed.) Encyclopedia of plant and crop science. Marcel Dekker, New York.

Karlen, D.L., E.C. Hurley, S.S. Andrews, C.A. Cambardella, D.W. Meek, M.D. Duffy, and A.P. Mallarino. 2006. Crop rotation effects on soil quality at three Northern Corn/Soybean Belt locations. Agron. J. 98:484–495.

Littell, R.C., G.A. Milliken, W.W. Stroup, and W.W. Wolfinger. 1996. SAS system for mixed models. SAS Institute, Cary, NC.

Magdoff, F., and H. van Es. 2000. Crop rotations. p. 99–108. *In* Building soils for better crops. 2nd ed. Sustainable Agric. Publ., Univ. of Vermont, Burlington.

Mannering, J.V., and D.R. Griffith. 1981. Value of crop rotation under various tillage systems. Agron. Guide AY-230. Coop. Ext. Serv., Purdue Univ., West Lafayette, IN.

Meese, B.G., P.R. Carter, E.S. Oplinger, and J.W. Pendleton. 1991. Corn/soybean rotation effect as influenced by tillage, nitrogen, and hybrid/cultivar. J. Prod. Agric. 4:74–80.

Oberle, S.L. 1994. Farming systems options for U.S. agriculture: An agroecological perspective. J. Prod. Agric. 7:119–123.

Oberle, S.L., and D.R. Keeney. 1990. Factors influencing corn fertilizer N requirements in the Northern Corn Belt. J. Prod. Agric. 3:527–534.

Pedersen, P., and J.G. Lauer. 2002. Influence of rotation sequence and tillage system on the optimum plant population for corn and soybean. Agron. J. 94:968–974.

Pedersen, P., and J.G. Lauer. 2003. Corn and soybean response to rotation sequence, row spacing, and tillage system. Agron. J. 95:965–971.

Peterson, T.A., and G.E. Varvel. 1989. Crop yield as affected by rotation and nitrogen rate. I. Soybean. Agron. J. 81:727–731.

Pikul, J.L., Jr., L. Hammack, and W.E. Riedell. 2005. Corn yield, nitrogen use, and corn rootworm infestation of rotations in the northern corn belt. Agron. J. 97:854–863.

Pimentel, D., J. Krummel, D. Gallahan, J. Hough, A. Merrill, I. Schreiner, P. Vittum, F. Koziol, E. Back, D. Yen, and S. Fiance. 1978. Benefits and costs of pesticide use in U.S. food production. Bioscience 28:772–784.

Porter, P.M., D.R. Huggins, C.A. Perillo, S.R. Quiring, and R.K. Crookston. 2003. Organic and other management strategies with two- and four-year crop rotations in Minnesota. Agron. J. 95:233–244.

Porter, P.M., J.G. Lauer, W.E. Lueschen, J.H. Ford, T.R. Hoverstad, E.S. Oplinger, and R.K. Crookston. 1997. Environment affects the corn and soybean rotation effect. Agron. J. 89:641–448.

Raimbault, B.A., and T.J. Vyn. 1991. Crop rotation and tillage effects on corn growth and soil structural stability. Agron. J. 83:979–985.

Randall, G.W. 2003. Present-day agriculture in southern Minnesota—Is it sustainable? Available at http://sroc.cfans.umn.edu/research/soils/publications/Present-Day%20Agriculture.pdf (verified 19 Feb. 2008). Univ. of Minnesota, Southern Res. and Outreach Ctr., Waseca, MN.

Raun, W.R., and G.V. Johnson. 1999. Improving nitrogen use efficiency for cereal production. Agron. J. 91:357–363.

SAS Institute. 2002. SAS release 9.1. The SAS Inst., Cary, NC.

Scharf, P.C., N.R. Kitchen, K.A. Sudduth, J.G. Davis, V.C. Hubbard, and J.A. Lory. 2005. Field-scale variability in optimal nitrogen fertilizer rate for corn. Agron. J. 97:452–461.

Tollenaar, M., and J. Wu. 1999. Yield improvement in temperate maize is attributable to greater stress tolerance. Crop Sci. 39:1597–1604.

USDA National Agricultural Statistics Service. 2004. Historical track records. Available at www.usda.gov/nass/pubs/trackrec/croptr04.pdf (verified 19 Feb. 2008). USDA-NASS, Washington, DC.

USDA National Agricultural Statistics Service. 2006a. Statistics of fertilizer and pesticides. Available at www.nass.usda.gov/Publications/Ag_Statistics/2006/CHAP14.PDF (verified 19 Feb. 2008). USDA-NASS, Washington, DC.

USDA National Agricultural Statistics Service. 2006b. Historical track records. Available at www.usda.gov/nass/pubs/trackrec/croptr06.pdf (verified 19 Feb. 2008). USDA-NASS, Washington, DC.

Vanotti, M.B., and L.G. Bundy. 1994a. An alternative rationale for corn nitrogen fertilizer recommendations. J. Prod. Agric. 7:243:249

Vanotti, M.B., and L.G. Bundy. 1994b. Corn nitrogen recommendations based on yield response data. J. Prod. Agric. 7:249–256.

Vanotti, M.B., and L.G. Bundy. 1994c. Frequency of nitrogen fertilizer carryover in the humid Midwest. Agron. J. 86:881–886.

Vanotti, M.B., and L.G. Bundy. 1995. Soybean effect on soil nitrogen availability in crop rotations. Agron. J. 86:676–680.

Welch, L.F. 1976. The Morrow plots: Hundred years of research. Ann. Agron. 27:881–890.

Yadav, S.N. 1997. Formulation and estimation of nitrate-nitrogen leaching from corn cultivation. J. Environ. Qual. 26:808–814.

Documento Dominio Music

Bassist Glenn Hughes Still Hoping to Perform with Deep Purple for Rock and Roll Hall of Fame Induction.

Although statements have been made to the contrary, former Deep Purple bassist Glenn Hughes still hopes to be able to perform when the group is inducted into the Rock and Roll Hall of Fame on April 8 at the Barclays Center in Brooklyn.

"All I know is I'm getting the award", Glenn Hughes, who played with the band for three studio albums between 1973-76, tells Billboard. "There's a lot of gossip and innuendo about who's saying what. I'd love to play. I'd love to sing. And so would David", referring to David Coverdale, who was part of Deep Purple's Mk. 3 (third iteration) and Mk. 4 (fourth iteration) lineups with Hughes.

The situation as it stands now, according to a statement from frontman Ian Gillan, is that while eight members from the band's first three lineups are set to be inducted, only the current lineup of the band -- Gillan, founding drummer Ian Paice, Mk. 2 bassist Roger Glover, guitarist Steve Morse and keyboardist Don Airey -- will perform during the ceremony. Gillan, Paice and Glover are perturbed that Morse and Airey, who have been with Deep Purple since 1994 and 2001 respectively, are not being inducted. Gillan's statement thanks Coverdale for understanding the situation, and Hughes is on board as well.

"A month ago David said to me, 'Shall I reach out to Ian Gillan to figure out what we're gonna do?' "Hughes recalls". He came back and told me that there's a problem that Don and Steve are not getting inducted and there's all sorts of scenarios with that and grumblings and this and that, and I said, 'Well, I'm just gonna stay out of the way', and Coverdale is as well. If Ian Gillan wants to run the show on behalf of his Deep Purple that's his business; my business is to show up and gracefully accept my award. But we're really hoping that we will be invited to sing. I'd like to think that Deep Purple can be just one big happy family on the night, y'know? Egos outside the door and be graceful in what we do."

Hughes has also directed his management to reach out to founding Purple guitarist Ritchie Blackmore, who's announced that he plans to skip the induction. "You just never know with him, but I'd love it if he showed up", Hughes says. "I just hope he has the balls to (attend), and for God's sake he bloody wrote those songs. He wrote those riffs. He wrote 'Smoke on the Water'. Whatever happened -- the eccentricities, the behavior, the name-calling and all that stuff -- I say let it all go. Blackmore should be there accepting his award. I'd be very upset if he didn't". Hughes adds that there were overtures to reunite the Mk. 3 lineup of Deep Purple before keyboardist Jon Lord passed away in 2012 "but we couldn't get Ritchie on the phone."

Hughes -- who was in the band Trapeze before joining Deep Purple and was also briefly with Black Sabbath and then Black Country Communion and California Breed -- is gearing up for a solo tour that will kick off Aug. 9 in Annapolis, Md., a start that was delayed to allow him to recover from double knee replacement surgery during January. "The knees are doing incredible", Hughes reports. "The knees are considered recovered now, which means the really extensive therapy has paid off. I mean, I couldn't walk at all. I was really, really in trouble, but now I'm good and looking forward to getting on stage again". Hughes is also planning to record a solo album during June that he hopes to have out by fall.

Documento Dominio Air Security

Australian Government
Department of Infrastructure, Transport, Regional Development and Local Government

Review of Aviation Security Screening: Report

Executive Summary

More than most other nations, Australia's aviation industry plays a critical role in the nation's economy and connectivity. The number of passenger movements through all Australian airports is forecast to increase by 4 per cent per annum over the next 20 years. As a result, passenger movements are expected to double by 2025–26 so that approximately 63 million passengers will be moving through Sydney Airport, 46.4 million through Melbourne Airport, 30 million through Brisbane Airport, 17.7 million through Perth Airport and 11.7 million through Adelaide Airport. With this projected growth, the challenge for Government and industry will be to ensure that the required security outcomes are achieved in a cost-effective manner while the movement of people and their baggage to their aircraft is facilitated through screening points in a timely and dignified manner.

The continued success of Australia's aviation industry will be driven in large measure by public confidence in the safety and security of regular public transport air services. To safeguard against unlawful interference with aviation, a layered approach to preventive security is underpinned by the *Aviation Transport Security Act 2004*. It includes measures for passenger and checked baggage screening. Screening is the process by which authorised screening officers inspect individuals and their property to deter and prevent the carriage of weapons or items that are considered to be a threat to an aircraft. In Australia, the Aviation Transport Security Regulations 2005 prescribe the type of items that are prohibited.

To ensure Australia is at the leading edge of screening systems and methodologies into the future, the Australian Government initiated a comprehensive Review of Aviation Security Screening in late 2007. The Review had several catalysts: the need to analyse and review the legislative framework; the growth in tourism and the aviation sector; expansion of the security regime to smaller industry participants; labour market pressures on the recruitment and retention of screening staff; and more recent legislative changes such as the introduction of liquids, aerosols and gels screening and the introduction of 100 per cent checked baggage screening to domestic airports.

In March 2008, the Minister for Infrastructure, Transport, Regional Development and Local Government, the Hon Anthony Albanese MP, appointed an external advisory group comprised of aviation industry leaders to guide the Review.

The Review examined a wide range of issues affecting aviation security screening, including the purpose of screening, service delivery and performance, national consistency, passenger experience, human factors, screening point design, the regulatory environment and the role of various technologies in the screening process. The Review identified several issues that impinge upon the effectiveness and efficiency of aviation security screening in Australia and makes 27 recommendations for improvements to screening.

At the core of these recommendations is the notion that aviation security screening must continually improve to ensure that it is effective, efficient and sustainable as Australia's aviation industry continues to grow.

For this to occur, all stakeholders—government, industry and the public—must clearly understand the purpose of screening. The Review has concluded that a definition of the purpose of screening could be articulated in a more easily understood form.

Documento Dominio Tourism

Introducing Ireland

Few countries have a tourist image so plagued by cliche as Ireland. From shamrocks and shillelaghs to leprechauns, lovable rogues and 40 shades of green, there's a plethora of platitudes to wade through before you scramble ashore on the real Ireland.

But it's well worth looking beyong the tourist tat, for the Emerald Isle (oops, there we go again) is one of Europe's gems, a scenic extravaganza of lake, mountain, sea and sky that's still gorgeous enough to make your jaw drop despite the best efforts of developers to scar some of the most beautiful bits with serried ranks of holiday homes. From the lonely, wind-lashed wilderness of Donegal to the picture-postcard harbour villages of Country Cork, there are countless opportunities to get outdoors and explore, whether it's surfing the beach breaks of Bundoran, cycling the coast of Country Antrim, or hiking the hills of Kerry and Connemara.

There are cultural pleasures too, of course, in the land of Joyce and Yeats, U2 and the Undertones. Dublin, Cork and Belfast all have top-notch restaurants, party-on pubs and a foot-stomping live-music scene, while you can track down impromptu pub sessions of traditional Irish music in places like Galway, Doolin and Killarney. And there's a wealth of history to discover, from the countless medieval castles and early Christian monasteries to the powerful political murals of Belfast and Derry, and one of the biggest concentrations of prehistoric monuments in Europe.

Documento Dominio Legal (Extensión 4 páginas)

Court of Appeal, Fourth District, Division 2, California.

IN RE: the MARRIAGE OF Dennis and Nancy SCHEPPERS. Dennis Scheppers, Appellant, v. Nancy Scheppers, Respondent.

No. E025054.

Decided: January 26, 2001

Dennis Scheppers, in pro. per., for Appellant. Janet Stouder Brandon, Riverside, for Respondent. Downey, Brand, Seymour & Rohwer, Mary J. Martinelli and Frank E. Dougherty, Folsom, for Association of Certified Family Law Specialists as Amicus Curiae upon the request of the Court of Appeal.

OPINION

A father of minor children appeals from an order modifying his child-support obligation. We affirm.

FACTUAL AND PROCEDURAL BACKGROUND

Dennis Scheppers and Nancy Scheppers married in 1974. The marriage produced six children: Micah, Matthew, Amber, Joseph, Amanda, and Jennifer. When their marriage was dissolved in 1987, all six children were minors.

In July of 1998, the mother applied for and obtained an order to show cause seeking, inter alia, a modification of child support for the two minor children living with her. Following an evidentiary hearing, the trial court set the father's child support obligation at $2,991 per month. The father appeals.

CONTENTIONS

In a somewhat different order, the father contends that the trial court erred in four respects: by failing to include in the mother's gross income sums she received as the beneficiary of a life insurance policy; by failing to impute any income to the mother based upon her ability to earn; by basing the father's income upon an unreasonable work schedule; and by depriving the father of due process and equal protection of the law.

ANALYSIS

A. THE TRIAL COURT DID NOT ERR BY EXCLUDING THE LIFE INSURANCE PROCEEDS FROM THE MOTHER'S INCOME.

Micah, the eldest child, committed suicide in February of 1998, when he was 22 years old. In February or March of that year, the mother received $200,568 as the beneficiary of an insurance policy insuring Micah's life. The father argued that those life insurance proceeds should be counted as income to the mother in 1998. The trial court decided that the insurance proceeds were an asset, not income. However, the court did include as income the interest that could be earned from the investment of that corpus. Finding that a reasonable rate of return was ten percent, the court deemed the mother to receive $1,666 interest income per month. On appeal, the father contends that the trial court erred by not treating the corpus of the life insurance death benefit as income.

The computation of the extent of a parent's obligation to support his or her child begins with the parent's annual gross income. "The annual gross income of each parent means income from whatever source derived, except as specified in subdivision (c) and includes, but is not limited to, the following: [¶] (1) Income such as commissions, salaries, royalties, wages, bonuses, rents, dividends, pensions, interest, trust income, annuities, workers' compensation benefits, unemployment insurance benefits, disability insurance benefits, social security benefits, and spousal support actually received from a person not a party to the proceeding to establish a child support order under this article. [¶] (2) Income from the proprietorship of a business, such as gross receipts from the business

reduced by expenditures required for the operation of the business. [¶] (3) In the discretion of the court, employee benefits or self-employment benefits, taking into consideration the benefit to the employee, any corresponding reduction in living expenses, and other relevant facts." (Fam.Code, § 4058, subd. (a).) 1 The only statutory exceptions are (1) "income derived from child support payments actually received," (2) "income derived from any public assistance program, eligibility for which is based on a determination of need," and (3) "[c]hild support received by a party for children from another relationship." (Id., subd. (c).)

Significantly, life insurance proceeds are not among the types of payments specifically included within the scope of the statutory definition. (§ 4058, subd. (a).) Nor are they specifically excluded. (Id., subd. (c).) The Legislature having failed to resolve the issue expressly, it falls to us to determine whether life insurance proceeds fall within the scope of the statutory definition of gross income.

Although the statutory definition is very broad (In re Marriage of Rocha (1998) 68 Cal.App.4th 514, 516, 80 Cal.Rptr.2d 376; Stewart v. Gomez (1996) 47 Cal.App.4th 1748, 1753, 1755, 55 Cal.Rptr.2d 531), it is not unlimited. It does not extend to every type of payment or economic benefit received by a parent. For instance, in addition to the statutory exceptions (§ 4058, subd. (c)), we have previously held that the proceeds of student loans are not income (In re Marriage of Rocha, supra, 68 Cal.App.4th at pp. 516-518, 80 Cal.Rptr.2d 376). Similarly, we conclude that life insurance benefits are not within the statutory definition of income, for the following reasons.

First, it is established that gifts, whether inter vivos (In re Marriage of Schulze (1997) 60 Cal.App.4th 519, 529, 70 Cal.Rptr.2d 488) or testamentary (County of Kern v. Castle (1999) 75 Cal.App.4th 1442, 1448-1454, 89 Cal.Rptr.2d 874), are not within the scope of the statutory definition of income. It is impossible to draw a rational distinction between a gift made by designating the donee as the beneficiary of a will and a gift made by designating the donee as the beneficiary of a life insurance policy.

Second, section 4058 refers to a variety of specific types of insurance benefits, including worker's compensation insurance, unemployment insurance, and disability insurance. (Id., subd. (a).) Clearly, the general category of insurance proceeds was before the Legislature. Had it intended to include life insurance proceeds, it presumably would have done so. Under those circumstances, it is unlikely that the omission of those benefits was unintentional.

Third, life insurance death benefits are not income under the federal Internal Revenue Code: "Except as otherwise provided in paragraph (2), subsection (d), and subsection (f), gross income does not include amounts received (whether in a single sum or otherwise) under a life insurance contract, if such amounts are paid by reason of the death of the insured." (26 U.S.C. § 101(a)(1).) Although federal law is not conclusive on the interpretation of section 4058, it is persuasive, because "[t]he operative language in subdivision (a) [of section 4058], i.e., 'annual gross income . means income from whatever source derived,' was lifted straight from the definition of income in section 61 of the Internal Revenue Code." (In re Marriage of Schulze, supra, 60 Cal.App.4th at p. 529, 70 Cal.Rptr.2d 488.)

Fourth, life insurance proceeds do not meet the common-law definition of income. The traditional understanding of "income" is the gain or recurrent benefit that is derived from labor, business, or property (McCulloch v. Franchise Tax Bd. (1964) 61 Cal.2d 186, 192, 37 Cal.Rptr. 636, 390 P.2d 412) or from any other investment of capital (Wells v. Wells (1944) 64 Cal.App.2d 113, 115-116, 148 P.2d 126). Almost every type of income specified by section 4058, subdivision (a), is either a return from labor, business, or property (such as wages, dividends, and rents) or else a substitute for that return (such as disability insurance benefits). Not only is a lump-sum life insurance death benefit not "recurrent," it is not derived from labor, business, or property in the same manner as the statutory examples.2

Fifth, including a lump-sum life insurance death benefit as income is impractical. If the mother were deemed to

have received $200,000 in income in the year in which the insurance proceeds were paid, what would happen the following year when her income would be $200,000 less? Would she be entitled to immediately move for an increase in child support? And if so, what is the sense in treating the insurance proceeds as income in the first place?

Finally, the only other court that we have found that has considered the matter has concluded that life insurance death benefits are not income. The Louisiana Court of Appeal, applying a statutory definition identical to section 4058 in all material respects, has held that life insurance proceeds should not be included as gross income for the purpose of determining the extent of the beneficiary's child support obligation. (Guy v. Guy (La.Ct.App.1992) 600 So.2d 771, 772.) The court's reason is persuasive: "Income is the key factor in our system, not capital or net worth. As stated in French [v. Wolf (1935) 181 La. 733, 737-738, 160 So. 396, 397], these concepts are distinctly different. Proceeds paid on a life insurance policy constitute capital and not income." (600 So.2d at p. 773.)

For all these reasons, we hold that life insurance death benefits are not within the scope of gross income as defined in section 4058.

The authorities that the father relies upon for the opposite conclusion are not persuasive. For instance, he notes that in County of Contra Costa v. Lemon (1988) 205 Cal.App.3d 683, at pages 688-689, 252 Cal.Rptr. 455, the court held that under the facts of that case, lottery winnings were properly included as income. But as the return on the investment of capital, gambling winnings fall within the traditional concept of income. (Wells v. Wells, supra, 64 Cal.App.2d at pp. 115-116, 148 P.2d 126.) Similarly, gambling winnings constitute taxable income under both federal and California law. (26 U.S.C. § 74(a) [prizes are income]; id., § 165(d) [gambling losses are deductible from income]; Rev. & Tax.Code, §§ 17071, 17081 [adopting federal provisions]; Campodonico v. United States (9th Cir.1955) 222 F.2d 310, 314 [gambling winnings constitute income].) Therefore, the conclusion that gambling winnings are income under section 4058 does not support the proposition that life insurance death benefits should also be considered to be income.

Moreover, even if lottery winnings did not fall within the general definition of income and were not treated as income under the tax laws, the rule of County of Contra Costa v. Lemon, supra, 205 Cal.App.3d 683, 252 Cal.Rptr. 455, has been limited to cases in which a county is seeking to recover reimbursement for public support paid under the Aid to Families with Dependent Children program or when failing to consider the lottery winnings would lead to a support order that is less than the minimum AFDC grant. (County of Kern v.. Castle, supra, 75 Cal.App.4th at pp. 1450-1451, 89 Cal.Rptr.2d 874.) Because there is no evidence that the mother here has ever collected AFDC payments, the rule of Lemon has no application. The federal cases cited by the father are similarly distinguishable because they both deal with the effect of certain types of payments on the recipient's eligibility for AFDC benefits. (Lukhard v. Reed (1987) 481 U.S. 368, 107 S.Ct. 1807, 95 L.Ed.2d 328 [personal injury awards]; LaMadrid v. Hegstrom (9th Cir.1987) 830 F.2d 1524 [personal injury awards, life insurance proceeds, worker's disability compensation, crime victim compensation].)

The trial court did not err by excluding from the calculation of the mother's gross income the principal sum that she received as a death benefit under her son's life insurance policy.

B.-D.**

DISPOSITION

The judgment is affirmed. Nancy Scheppers shall recover her costs on appeal.

FOOTNOTES

1. Unless specified otherwise, all further section references are to this code.

2. The amicus curiae raises the possibility that insurance policy death benefits would be analogous to a return on an investment if the policy premiums were paid by the beneficiary. However, we need not decide whether the proceeds would be income under those circumstances, because there is no evidence in the record that the mother paid the premiums for the policy in question. To the contrary, her counsel stated without contradiction in open court that the mother had received the life insurance proceeds "as a gift from her son," suggesting that the decedent paid his own premiums. Her trial brief is similar ("posthumous gift"). Because the decedent was in the United States Marine Corps at the time of his death, the evidence suggests that the policy was obtained by the decedent through his military service.

FOOTNOTE. See footnote *, ante.

McKINSTER, J.

RAMIREZ, P.J., and GAUT, J., concur.

Documento Dominio Twitter

@stellargirl I loooooooovvvvvveee my Kindle2. Not that the DX is cool, but the 2 is fantastic in its own right.
Reading my kindle2... Love it... Lee childs is good read.
Ok, first assesment of the #kindle2 ...it fucking rocks!!!
@kenburbary You'll love your Kindle2. I've had mine for a few months and never looked back. The new big one is huge! No need for remorse! :)
@mikefish Fair enough. But i have the Kindle2 and I think it's perfect :)
@richardebaker no. it is too big. I'm quite happy with the Kindle2.
Fuck this economy. I hate aig and their non loan given asses.
Jquery is my new best friend.
Loves twitter
how can you not love Obama? he makes jokes about himself.
Check this video out -- President Obama at the White House Correspondents' Dinner http://bit.ly/IMXUM
@Karoli I firmly believe that Obama/Pelosi have ZERO desire to be civil. It's a charade and a slogan, but they want to destroy conservatism
House Correspondents dinner was last night whoopi, barbara &amp
Watchin Espn..Jus seen this new Nike Commerical with a Puppet Lebron..sh*t was hilarious...LMAO!!!
dear nike, stop with the flywire. that shit is a waste of science. and ugly. love, @vincentx24x
#lebron best athlete of our generation, if not all time (basketball related) I don't want to get into inter-sport debates about __1/2
I was talking to this guy last night and he was telling me that he is a die hard Spurs fan. He also told me that he hates LeBron James.
i love lebron. http://bit.ly/PdHur
@ludajuice Lebron is a Beast, but I'm still cheering 4 the A..til the end.
@Pmillzz lebron IS THE BOSS
@sketchbug Lebron is a hometown hero to me, lol I love the Lakers but let's go Cavs, lol
lebron and zydrunas are such an awesome duo
@wordwhizkid Lebron is a beast... nobody in the NBA comes even close.
downloading apps for my iphone! So much fun :-) There literally is an app for just about anything.
good news, just had a call from the Visa office, saying everything is fine.....what a relief! I am sick of scams out there! Stealing!
http://twurl.nl/epkr4b - awesome come back from @biz (via @fredwilson)
In montreal for a long weekend of R&amp
Booz Allen Hamilton has a bad ass homegrown social collaboration platform. Way cool! #ttiv
[#MLUC09] Customer Innovation Award Winner: Booz Allen Hamilton -- http://ping.fm/c2hPP
@SoChi2 I current use the Nikon D90 and love it, but not as much as the Canon 40D/50D. I chose the D90 for the video feature. My mistake.
need suggestions for a good IR filter for my canon 40D ... got some? pls DM
@surfit: I just checked my google for my business- blip shows up as the second entry! Huh. Is that a good or ba... ? http://blip.fm/~6emhv
@phyreman9 Google is always a good place to look. Should've mentioned I worked on the Mustang w/ my Dad, @KimbleT.

Bibliografía

[1] S. J. Rusell and P. Norvig, *Inteligencia Artificial. Un enfoque Moderno*, 2nd ed. España: Pearson Prentice Hall, 2004.

[2] T. Berners-Lee, J. Hendler, O. Lassila, and others, “The semantic web,” *Scientific american*, vol. 284, no. 5, pp. 28–37, 2001.

[3] A. Gómez Pérez, O. Corcho, and M. Fernández López, *Ontological Engineering*, 1st ed. London: Springer-Verlag, 2004.

[4] G. Petasis, V. Karkaletsis, G. Paliouras, A. Krithara, and E. Zavitsanos, “Ontology Population and Enrichment: State of the Art,” in *Knowledge-Driven Multimedia Information Extraction and Ontology Evolution*, G. Paliouras, C. D. Spyropoulos, and G. Tsatsaronis, Eds. Springer Berlin Heidelberg, 2011, pp. 134–166.

[5] G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988.

[6] J. E. Gómez Balderas, J. Á. Vera Félix, and O. A. Olivas Zazueta, “Métodos Estadísticos en Procesamiento de Lenguaje Natural y su uso en Alineación de los Corpus Paralelos,” Instituto Politécnico Nacional. Centro de Investigación en Computación, Ciudad de México, Tech. Rep. 217, 2006.

[7] H. Cunningham, “Information Extraction, Automatic,” in *Encyclopedia of Language & Linguistics*, 2nd ed., Oxford: Elsevier, 2006, pp. 665–677.

[8] R. Mitkov, *The Oxford Handbook of Computational Linguistics*, 1st ed. Oxford, Great Britain: Oxford University Press, 2003.

[9] A. Clark, C. Fox, and S. Lappin, Eds., *The handbook of computational linguistics and natural language processing*, vol. 57. Malden, MA, United States: WILEY-BLACKWELL, 2010.

[10] P. Kordjamshidi and M.-F. Moens, “Global machine learning for spatial ontology population,” *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 30, pp. 3–21, 2015.

[11] P. P. Talukdar, J. Reisinger, M. Paşca, D. Ravichandran, R. Bhagat, and F. Pereira, “Weakly-supervised acquisition of labeled class instances using graph random walks,” in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Seattle, 2008, pp. 582–590.

[12] D. A. De Araujo, S. J. Rigo, C. Muller, and R. Chishman, “Automatic Information Extraction from Texts with Inference and Linguistic Knowledge Acquisition Rules,” in *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, Atlanta, 2013, vol. 3, pp. 151–154.

[13] C. Faria, I. Serra, and R. Girardi, “A domain-independent process for automatic ontology population from text,” *Science of Computer Programming*, vol. 95, pp. 37–50, 2013.

[14] R. Lima, H. Oliveira, F. Freitas, and B. Espinasse, “Ontology Population from the Web: An Inductive Logic Programming-Based Approach,” in *2014 11th International Conference on Information Technology: New Generations (ITNG)*, Las Vegas, 2014, pp. 473–478.

[15] J. Contreras, “Incremento crítico del conocimiento de la Web semántica mediante poblado automático de ontologías,” Tesis Doctoral, Facultad de Informática Universidad Politécnica de Madrid, Madrid, 2004.

[16] G. Geleijnse and J. H. Korst, “Automatic Ontology Population by Googling.,” in *Proceedings of the 17th Belgium-Netherlands Conference on Artificial Intelligence*, Brussels, 2005, pp. 120–126.

[17] C. Faria, R. Girardi, and P. Novais, “Using domain specific generated rules for automatic ontology population,” in *2012 12th International Conference on Intelligent Systems Design and Applications (ISDA)*, Kochi, 2012, pp. 297–302.

[18] M. Pasca, “Acquisition of categorized named entities for web search,” in *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, New York, 2004, pp. 137–145.

[19] W. IJntema, J. Sangers, F. Hogenboom, and F. Frasincar, “A lexico-semantic pattern language for learning ontology instances from text,” *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 15, pp. 37–50, 2012.

[20] L. J. Nederstigt, S. S. Aanen, D. Vandic, and F. Frasincar, “FLOPPIES: A Framework for Large-Scale Ontology Population of Product Information from Tabular Data in E-commerce Stores,” *Decision Support Systems*, vol. 59, pp. 296–311, 2014.

[21] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan, “GATE: an architecture for development of robust HLT applications,” in *Proceedings of the 40th annual meeting on association for computational linguistics*, Stroudsburg, PA, USA, 2002, pp. 168–175.

[22] G. Luger, *Artificial intelligence: structures and strategies for complex problem solving*, 5th ed. Edinburgh Gate, England: Addison - Wesley, 2005.

[23] P. Ponce Cruz, *INTELIGENCIA ARTIFICIAL CON APLICACIONES A LA INGENIERÍA*, 1st ed. México: Alfaomega, 2010.

[24] T. Gruber, “A translation approach to portable ontology specifications,” *Knowledge Acquisition*, vol. 5, no. 2, pp. 199–220, 1993.

[25] W. Borst, *Construction of Engineering Ontologies*. Enschede, The Netherlands: Centre for Telematica and Information Technology, University of Tweenty., 1997.

[26] S. Staab and R. Studer, *Handbook on ontologies*, 2nd ed. London: Springer, 2004.

[27] N. Guarino, “Semantic matching: Formal ontological distinctions for information organization, extraction, and integration,” in *Information Extraction A Multidisciplinary Approach to an Emerging Information Technology*, M. T. Pazienza, Ed. Springer Berlin Heidelberg, 1997, pp. 139–170.

[28] M. Khosrow-Pour, *Dictionary of Information Science and Technology*. Hershey, PA: IGI Global, 2006.

[29] N. F. Noy, R. W. Fergerson, and M. A. Musen, "The Knowledge Model of Protégé-2000: Combining Interoperability and Flexibility," in *Knowledge Engineering and Knowledge Management Methods, Models, and Tools*, R. Dieng and O. Corby, Eds. Springer Berlin Heidelberg, 2000, pp. 17–32.

[30] H. Knublauch, R. W. Fergerson, N. F. Noy, and M. A. Musen, "The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications," in *The Semantic Web – ISWC 2004*, S. A. McIlraith, D. Plexousakis, and F. van Harmelen, Eds. Springer Berlin Heidelberg, 2004, pp. 229–243.

[31] L. Yu, "Jena: A Framework for Development on the Semantic Web," in *A Developer's Guide to the Semantic Web*, Springer Berlin Heidelberg, 2011, pp. 491–532.

[32] M. Uschold and R. Jasper, "A framework for understanding and classifying ontology applications," in *Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods (KRR5), Stockholm, Sweden*, Stockholm, Sweden, 1999, pp. 12–24.

[33] P. Cimiano, *Ontology learning and population from text - algorithms, evaluation and applications.* Karlsruhe Germany: Springer US, 2006.

[34] W. Wong, W. Liu, and M. Bennamoun, "Ontology Learning from Text: A Look Back and into the Future," *ACM Comput. Surv.*, vol. 44, no. 4, p. 20:1–20:36, 2012.

[35] N. O. Garanina and E. A. Sidorova, "Ontology population as algebraic information system processing based on multi-agent natural language text analysis algorithms," *Program Comput Soft*, vol. 41, no. 3, pp. 140–148, 2015.

[36] S. Sarawagi, "Information extraction," *Foundations and trends in databases*, vol. 1, no. 3, pp. 261–377, 2008.

[37] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investigationes*, vol. 30, no. 1, pp. 3–26, 2007.

[38] J. Shafi and A. Ali, "Defining Relations in Precisiation of Natural Language Processing for Semantic Web," *International Journal on Computer Science & Engineering*, vol. 4, no. 5, pp. 723–728, 2012.

[39] C. Zapata, K. Palomino, and R. Rosero, "A Method for coordinative and prepositional syntactic disambiguation," *DYNA*, vol. 75, no. 156, pp. 29–42, 2008.

[40] S. Lappin, C. Ebert, H. Gregory, and N. Nicolov, "Full Paraphrase Generation for Fragments in Dialogue," in *Current and New Directions in Discourse and Dialogue*, J. van Kuppevelt and R. W. Smith, Eds. Springer Netherlands, 2003, pp. 161–181.

[41] A. Copestake, D. Flickinger, C. Pollard, and I. A. Sag, "Minimal Recursion Semantics: An Introduction," *Research Language Computation*, vol. 3, no. 2–3, pp. 281–332, 2005.

[42] S. N. Galicia H. and A. Gelbukh, *Investigaciones en análisis sintáctico para el español*, 1st ed. México: Instituto Politécnico Nacional, Dirección de Publicaciones, 2007.

[43] J. F. Sowa, "Conceptual graphs," in *Handbook of Knowledge Representation*, 2008, pp. 213--237.

[44] R. Danger, “Extraccion y análisis de información desde la perspectiva de la Web Semántica,” Tesis Doctoral, Universitat Jaume II, Castellón, España, 2007.

[45] W. Duch, “Rule-Based Methods,” *Department of Informatics, Nicolaus Copernicus University, Poland*, 2010.

[46] K. Siau and M. Rossi, “Evaluation of information modeling methods-a review,” in *, Proceedings of the Thirty-First Hawaii International Conference on System Sciences*, Kohala Coast, HI, 1998, vol. 5, pp. 314–322.

[47] G. Shanks, E. Tansley, and R. Weber, “Using ontology to validate conceptual models,” *Communications of the ACM*, vol. 46, no. 10, pp. 85–89, 2003.

[48] D. M. Powers, “Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation,” *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[49] C. Faria and R. Girardi, “An Information Extraction Process for Semi-automatic Ontology Population,” in *Soft Computing Models in Industrial and Environmental Applications, 6th International Conference SOCO 2011*, Springer Berlin Heidelberg, 2011, pp. 319–328.

[50] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python*, 1st ed. United States of America: O’Reilly Media, Inc., 2009.

[51] H. Cunningham *et al.*, *Developing Language Processing Components with GATE Version 8.1:(a User Guide)*. Sheffield: University of Sheffield Department of Computer Science, 2015.

[52] A. Kiryakov, D. Ognyanov, and D. Manov, “OWLIM – A Pragmatic Semantic Repository for OWL,” in *Web Information Systems Engineering – WISE 2005 Workshops*, M. Dean, Y. Guo, W. Jun, R. Kaschek, S. Krishnaswamy, Z. Pan, and Q. Z. Sheng, Eds. Springer Berlin Heidelberg, 2005, pp. 182–192.

[53] R. J. Abbott, “Program design by informal English descriptions,” *Communications of the ACM*, vol. 26, no. 11, pp. 882–894, 1983.

[54] V. de Boer, M. van Someren, and B. J. Wielinga, “A redundancy-based method for the extraction of relation instances from the Web,” *International Journal of Human Computer Studies*, vol. 65, no. 9, pp. 816–831, 2007.

[55] H.-G. Yoon, Y. J. Han, S.-B. Park, and S.-Y. Park, “Ontology Population from Unstructured and Semi-structured Texts,” in *Sixth International Conference on Advanced Language Processing and Web Information Technology, 2007. ALPIT 2007*, Luoyang, Henan, 2007, pp. 135–139.

[56] A. P. Manine, E. Alphonse, and P. Bessieres, “Information Extraction as an Ontology Population Task and Its Application to Genic Interactions,” in *20th IEEE International Conference on Tools with Artificial Intelligence, 2008. ICTAI ’08*, Dayton, OH, 2008, vol. 2, pp. 74–81.

[57] J. M. Ruiz-Martinez *et al.*, “Populating Ontologies in the eTourism Domain,” in *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT ’08*, Sydney, NSW, 2008, vol. 3, pp. 316–319.

[58] R. Danger and R. Berlanga, “Generating complex ontology instances from documents,” *Journal of Algorithms*, vol. 64, no. 1, pp. 16–30, 2009.

[59] A. Schlaf and R. Remus, “Learning Categories and their Instances by Contextual Features,” in *Proceedings of the 8th International Conference on Language Resources and Evaluation, LREC*, Istanbul, 2012, pp. 1235–1239.

[60] J. M. Ruiz-Martínez, R. Valencia-García, R. Martínez-Béjar, and A. Hoffmann, “BioOntoVerb: A top level ontology based framework to populate biomedical ontologies from texts,” *Knowledge-Based Systems*, vol. 36, pp. 68–80, 2012.

[61] D. Sadoun, C. Dubois, Y. Ghamri-Doudane, and B. Grau, “From Natural Language Requirements to Formal Specification Using an Ontology,” in *2013 IEEE 25th International Conference on Tools with Artificial Intelligence (ICTAI)*, Herndon, 2013, pp. 755–760.

[62] A. B. Ríos, “Obtención de axiomas en el aprendizaje de ontologías,” Tesis Doctoral, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, Victoria, Tamaulipas, México, 2013.

[63] F. Colace, M. De Santo, L. Greco, F. Amato, V. Moscato, and A. Picariello, “Terminological ontology learning and population using latent Dirichlet allocation,” *Journal of Visual Languages & Computing*, vol. 25, no. 6, pp. 818–826, 2014.

[64] S. Santos and R. Girardi, “Apponto-Pro: An incremental process for ontology learning and population,” in *2014 9th Iberian Conference on Information Systems and Technologies (CISTI)*, Barcelona, 2014, pp. 1–6.

[65] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, “Systematic literature reviews in software engineering–a systematic literature review,” *Information and software technology*, vol. 51, no. 1, pp. 7–15, 2009.

[66] R. J. Wieringa, *Design science methodology for information systems and software engineering*. Berlin: Springer, 2014.

[67] C. M. Zapata J., “Definición de un esquema preconceptual para la obtención automática de esquemas conceptuales de UML,” Ph.D Thesis, UNAL, Medellín, 2007.

[68] A. Taylor, M. Marcus, and B. Santorini, “The Penn treebank: an overview,” in *Treebanks*, vol. 20, Springer Science, 2003, pp. 5–22.

[69] J. Payne, R. Huddleston, and G. K. Pullum, “The distribution and category status of adjectives and adverbs,” *Word Structure*, vol. 3, no. 1, pp. 31–81, 2010.

[70] R. Ribeiro de Azevedo, F. Freitas, R. G. C. Rocha, J. A. Alves de Menezes, C. M. de Oliveira Rodrigues, and G. D. F. P. E. Silva, “An Approach for Learning and Construction of Expressive Ontology from Text in Natural Language,” in *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, Warsaw, 2014, vol. 1, pp. 149–156.