

**CONSTRUCCIÓN DE PROTOTIPOS PARA LAS ETAPAS DE UN
TRADUCTOR DE IDIOMAS BASADA EN EL ANÁLISIS Y
ESTRUCTURACIÓN DE UNA PLATAFORMA TEÓRICA**

**JOSÉ FERNEY FRANCO BAQUERO
CÉSAR FREDY GIL MEJÍA
SEBASTIÁN MARÍN LOAIZA**

**UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA, FÍSICA
Y DE SISTEMAS
PROGRAMA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
PEREIRA
2006**

**CONSTRUCCIÓN DE PROTOTIPOS PARA LAS ETAPAS DE UN
TRADUCTOR DE IDIOMAS BASADA EN EL ANÁLISIS Y
ESTRUCTURACIÓN DE UNA PLATAFORMA TEÓRICA**

**JOSÉ FERNEY FRANCO BAQUERO
CÉSAR FREDY GIL MEJÍA
SEBASTIÁN MARÍN LOAIZA**

Trabajo de grado

**Director
JOHN ALEXIS GUERRA GÓMEZ
Ingeniero de Sistemas y Computación**

**UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA, FÍSICA
Y DE SISTEMAS
PROGRAMA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
PEREIRA
2006**

Nota de aceptación:

Firma del presidente del jurado

Firma del jurado

Pereira, 7 de Julio de 2006

Para tí, madre, por la paciencia, la entereza, el ánimo que me das todos los días y el generoso amor que demuestras en tu sonrisa. Eres la razón por la que lucho insaciablemente, orgulloso con el sudor de mi frente.

A mi familia entera por estar tan pendientes de mis logros, a mi tío Alexis por sus contribuciones e incentivos para realizar mis estudios, a mi tía Gloria porque sé que este logro le alegra tanto como si fuera su hijo, a mi hermana Claudia porque me da miles de razones para sacar mi familia adelante, a Danielita por toda la alegría que me da, a Yuli por su apoyo y por ser mi cómplice, a mi abuela Elvia por todo el tiempo que ha esperado para celebrar conmigo este triunfo, a Darío por los enormes esfuerzos que hace para contribuir con el bienestar familiar, a todos ustedes, a mis amigos de toda la vida y a los que no nombré pero siempre están allí presentes para apoyarme, quiero que celebremos juntos la culminación de esta etapa que para muchos hoy termina, pero para mí es solo el comienzo de mejores cosas...

Fredy

A mi padre que me inculcó la importancia de estudiar y de formarme como persona, sin escatimar esfuerzos y brindándome su apoyo con alegría y desinterés. A mi madre cuyo sacrificio y amor fueron fundamentales para este logro. A mi familia y amigos.

Sebastián

Dedico este trabajo en primer lugar a mi padre, quien durante tantos años se ha esforzado por brindarme lo mejor, él desde mi niñez se preocupó por mi educación y por darme a través de todos los medios la forma de estudiar hasta lograr los resultados actuales. De igual manera a mi madre quien dio lo mejor de sí para hacer de mí un hombre bien y quien con su paciencia, cariño y sacrificio ha estado a mi lado en los momentos decisivos de mi vida. A mi hermano Fredy a quien admiro y aprecio y que de igual manera me ayudó a lo largo de mi carrera, que siempre confió en mí y con sus consejos ha enrutado mi vida académica. A mi familia, especialmente Shirley, a quien quiero como si fuese mi hermana, Divaldo mi amigo, mis tíos Ofir, Gabriel, Eulises, Lileardo, Myriam Quintero los cuales con su cariño, consejos y confianza en mí han aportado enormemente a mi vida. A amigos y compañeros, parte de este triunfo es de ellos. A los profesores que me han transmitido su conocimiento estoy ampliamente agradecido. A todos aquellos que contribuyeron en la culminación de esta etapa.

Ferney

AGRADECIMIENTOS

Para la realización del presente trabajo muchas personas han realizado contribuciones importantes, contribuciones sin las cuales tal vez nuestra tesis no hubiera podido realizarse.

En primer lugar agradecemos al ingeniero Omar Iván Trejos Buriticá quien desde un principio creyó en nuestra idea, nos animó y nos ayudó a darle forma al proyecto. Al licenciado Alexánder Quintero, el cual a través de su conocimiento y su amplia visión en el desarrollo de proyectos nos ayudó a colocar los primeros cimientos sobre los cuales se constituiría nuestra labor posterior, de igual manera por la revisión y correcciones realizadas al documento final. Al ingeniero John Alexis Guerra Gómez, nuestro asesor y compañero, sin el cual no hubiésemos culminado satisfactoriamente. Sus consejos y experiencia nos animaron a seguir adelante y a mirar nuestra investigación con mejores ojos. Al magíster y profesor Rafael Areiza Londoño, quien gentilmente nos proporcionó asesoría en un tema tan desconocido para nosotros como lo es la lingüística. Al ingeniero Gilberto Vargas Cano, por sus ideas y consejos iniciales. Al ingeniero César Augusto Castaño Salazar por su valiosa colaboración en la traducción y revisión de textos al inglés. Al grupo de desarrollo en Allegro GDA por permitirnos usar su servidor para nuestras labores y al semillero de investigación PULPA por acoger nuestro proyecto.

De igual manera a Microtec de Colombia Ltda. que facilitó sus instalaciones en repetidas ocasiones. A la familia Marín y la familia Franco quienes abrieron amablemente la puerta de sus hogares permitiéndonos trabajar allí.

Quisiéramos también agradecer al señor John W. Hutchins, experto en traducción automática quien se tomó el tiempo de leer y poner en consideración los resultados obtenidos en el presente trabajo. A los doctores españoles Jesús Vilares y Jorge Graña por el acceso permitido a nuestro grupo de investigación a sus herramientas de procesamiento y etiquetamiento. A los señores Steven Bird y Edwar Loper de la universidad de Pensilvania desarrolladores de la poderosa herramienta de procesamiento de lenguaje natural NLTK en la cual nos soportamos para el desarrollo de algunas de las etapas de nuestra investigación.

A todos aquellos que nos ayudaron e incentivaron a seguir adelante y a quienes de alguna manera contribuyeron al desarrollo del presente trabajo. Nuestros más profundos y sinceros agradecimientos.

CONTENIDO

	pág.
RESUMEN	15
INTRODUCCIÓN	16
1. FUNDAMENTOS GENERALES	17
1.1. TRADUCCIÓN AUTOMÁTICA	17
1.2. EL PROCESO DE TRADUCCIÓN	19
1.3. ETAPAS DEL PROCESO DE TRADUCCIÓN	20
1.3.1. Preprocesamiento	20
1.3.2. Tokenización	20
1.3.3. Análisis Morfológico	21
1.3.4. Análisis Sintáctico	21
1.3.5. Análisis Semántico	21
1.3.6. Síntesis al lenguaje destino	22
2. ESTADO DEL ARTE	23
2.1. LA NECESIDAD DE LA TRADUCCIÓN	23
2.1.1. El problema: Diversidad de idiomas	23
2.1.2. La solución: Traducción	24
2.2. EL SECTOR DE LA TRADUCCIÓN	26
2.2.1. Traducción	26
2.2.2. Revisión	26
2.2.3. Interpretación	26
2.2.4. Terminología	27
2.2.5. Localización	27
2.2.6. Edición	28
2.2.7. Traducción Audiovisual	29
2.3. ANÁLISIS DEL SECTOR DE LA TA	29
2.4. DIVISIÓN DE LOS SISTEMAS DE TA	31
2.4.1. Cantidad de idiomas soportados	31
2.4.2. Dirección de la traducción	31
2.4.3. Grado de Automatización	31
2.5. USO DE SISTEMAS DE TA	32
2.5.1. Diseminación	32
2.5.2. Asimilación	32
2.5.3. Intercambio	32
2.5.4. Acceso a Bases de Datos (BD)	32
2.6. ENFOQUES DE LA TA	33
2.6.1. Enfoques tradicionales a la TA	33
2.6.2. Enfoques modernos de TA	34
2.6.3. Representación del conocimiento	35

2.7.	HISTORIA DE LA TA	37
2.7.1.	Precursores y pioneros 1933 - 1954	39
2.7.2.	Primera década	39
2.7.3.	Segunda década	40
2.7.4.	Tercera década	42
2.7.5.	Cuarta década	43
2.7.6.	Quinta década	44
2.7.7.	Sexta década	45
2.7.8.	Historia y antecedentes de SYSTRAN	46
2.8.	LA TA A NIVEL MUNDIAL	47
2.8.1.	Sistemas de TA utilizados a nivel mundial	47
2.8.2.	Análisis del compendio de la EAMT de software de traducción	47
2.9.	DESCRIPCIÓN FUNCIONAL DE PRODUCTOS EXISTENTES	48
2.9.1.	Software de TA	48
2.9.2.	Herramientas para el soporte de traducción	49
2.9.3.	Servicios de traducción	52
2.10.	MERCADO DE LA TA	53
3.	PREPROCESAMIENTO	56
3.1.	ARQUITECTURA DEL PREPROCESADOR	56
3.1.1.	Filtro	56
3.1.2.	Contracciones	59
3.1.3.	Preprocesamiento dependiente del idioma	64
3.2.	USO DEL PREPROCESADOR	67
4.	TOKENIZACIÓN	69
4.1.	INTRODUCCIÓN	69
4.2.	ASPECTOS TEÓRICOS	69
4.2.1.	Consideraciones preliminares	69
4.2.2.	El proceso de tokenización	71
4.2.3.	Representación de tokens	74
4.3.	MODELAMIENTO DEL TOKENIZADOR	75
4.4.	IMPLEMENTACIÓN DEL TOKENIZADOR	77
5.	DICCIONARIO	84
5.1.	INTRODUCCIÓN	84
5.2.	ASPECTOS TEÓRICOS SOBRE LOS DICCIONARIOS	84
5.2.1.	Importancia de los diccionarios en la TA	84
5.2.2.	Diccionarios de papel	85
5.2.3.	Tipos de información de las palabras	85
5.3.	MODELAMIENTO DEL DICCIONARIO	87
5.3.1.	Características esenciales de la estructura del diccionario	87
5.3.2.	Estructuras Hash	88
5.3.3.	Diagrama del diccionario basado en Hash	88
5.4.	IMPLEMENTACIÓN DEL PROTOTIPO DE DICCIONARIO	91
5.4.1.	Representación de un diccionario en un sistema de TA	91

5.4.2.	Formato de entrada y salida	92
5.4.3.	Explicación del código fuente	94
6.	ANÁLISIS MORFOLÓGICO	103
6.1.	INTRODUCCIÓN	103
6.2.	ASPECTOS TEÓRICOS	104
6.2.1.	Lexemas	104
6.2.2.	Morfemas	104
6.2.3.	Clasificación de los tipos de estructuras morfológicas	105
6.2.4.	Proceso de análisis morfológico	107
6.2.5.	Otras consideraciones	111
6.3.	MODELAMIENTO DEL ANALIZADOR MORFOLÓGICO	112
6.4.	IMPLEMENTACIÓN DEL ANALIZADOR MORFOLÓGICO	114
7.	ANÁLISIS SINTÁCTICO	125
7.1.	INTRODUCCIÓN	125
7.2.	ASPECTOS TEÓRICOS	125
7.2.1.	Enfoques de estructuras gramaticales	125
7.2.2.	Parsing	128
7.3.	MODELAMIENTO DEL ANALIZADOR SINTÁCTICO	131
7.4.	IMPLEMENTACIÓN DEL ANALIZADOR SINTÁCTICO	132
7.4.1.	Ejemplos	136
8.	ANÁLISIS SEMÁNTICO	141
8.1.	INTRODUCCIÓN	141
8.2.	ASPECTOS TEÓRICOS	141
8.2.1.	Representación del conocimiento	141
8.3.	MODELAMIENTO DEL ANALIZADOR SEMÁNTICO	144
8.4.	IMPLEMENTACIÓN DEL ANALIZADOR SEMÁNTICO	148
8.4.1.	Funcionamiento	154
9.	SÍNTESIS AL LENGUAJE DESTINO	159
9.1.	INTRODUCCIÓN	159
9.2.	ASPECTOS TEÓRICOS	159
9.2.1.	El verbo	159
9.2.2.	La formación de plurales	162
9.2.3.	Cambio de género	163
9.2.4.	Terminación en or	164
9.3.	MODELAMIENTO DE LA SÍNTESIS	164
9.3.1.	Conjugación de Verbos	164
9.3.2.	Conformación de plurales	165
9.3.3.	Cambio de género	165
9.3.4.	Conjunción de nombres	165
9.4.	IMPLEMENTACIÓN DEL PROTOTIPO	166
9.4.1.	Plurales	166
9.4.2.	Géneros	168
9.4.3.	Conjugador	170

9.4.4.	Múltiples pronombres	174
9.4.5.	Función Principal	178
10.	PROTOTIPO GRÁFICO	181
11.	APORTES	194
11.1.	APORTES DEL DESARROLLO	194
11.2.	APORTES EN CUANTO A LA METODOLOGÍA DE TRABAJO	197
12.	CONCLUSIONES	199
13.	RECOMENDACIONES	202
14.	TRABAJO FUTURO	203
	BIBLIOGRAFÍA	205
	ÍNDICE	209
	ANEXOS	214

LISTA DE TABLAS

	pág.
Tabla 1 Clasificación de idiomas por número de parlantes	23
Tabla 2 Gramática definida para el inglés, frases afirmativas, presente simple	131
Tabla 3 Cambio de género para adjetivos, palabras agudas	164
Tabla 4 Correspondencia de persona y número	175
Tabla 5 Entrada y salida del proceso de síntesis	179

LISTA DE FIGURAS

	pág.
Figura 1 Proceso de traducción automática	19
Figura 2 Arquitectura del Preprocesador	56
Figura 3 Texto antes y después de filtrar	57
Figura 4 Texto antes y después de expandir contracciones	62
Figura 5 Texto antes y después de preprocesar contracciones posesivas	66
Figura 6 Modelamiento estructura del diccionario	89
Figura 7 Clasificación de los morfemas	105
Figura 8 Árbol Sintáctico "I play soccer and you play piano"	136
Figura 9 Árbol Sintáctico "NLTK uses a module for Tokenization"	138
Figura 10 Árbol Sintáctico "Gabriel Garcia Marquez is an important author"	139
Figura 11 Estructura gramatical básica de una frase	145
Figura 12 Árbol "I play soccer and she plays the pianos"	155
Figura 13 Árbol "The kid bleats and the kid speaks"	155
Figura 14 Árbol con roles "I play soccer and she plays the pianos"	156
Figura 15 Árbol con roles "The kid bleats and the kid speaks"	156
Figura 16 Árbol con roles no ambiguos "I play soccer and she plays the pianos"	157
Figura 17 Árbol con roles no ambiguos "The kid bleats and the kid speaks"	157
Figura 18 Prototipo gráfico - Ventana Inicial	181
Figura 19 Prototipo gráfico - Ventana de ayuda	182
Figura 20 Prototipo gráfico - Ventana principal	183
Figura 21 Prototipo gráfico - Entrada de texto	184
Figura 22 Prototipo gráfico - Ejemplo entrada desde archivo I	185
Figura 23 Prototipo gráfico - Ejemplo entrada desde archivo II	185
Figura 24 Prototipo gráfico - Diálogo para abrir archivo	186
Figura 25 Prototipo gráfico - Selección fuente de traducción	186
Figura 26 Prototipo gráfico - Menú proceso de traducción	187
Figura 27 Prototipo gráfico - Salida preprocesamiento	187
Figura 28 Prototipo gráfico - Salida tokenización	188
Figura 29 Prototipo gráfico - Salida análisis morfológico	189
Figura 30 Prototipo gráfico - Animación Parser recursivo descendente	190
Figura 31 Prototipo gráfico - Salida análisis sintáctico	191
Figura 32 Prototipo gráfico - Salida análisis semántico	191
Figura 33 Prototipo gráfico - Salida síntesis al lenguaje destino	192
Figura 34 Prototipo gráfico - Salida de la traducción	193

LISTA DE ANEXOS

	pág.
ANEXO A	TENDENCIAS DE IDIOMAS Y CULTURAS 214
ANEXO B	PROBLEMAS DE COMUNICACIÓN EN GRANDES EMPRESAS 217
ANEXO C	ASPECTOS CULTURALES 221
ANEXO D	DATOS ESTADISTICOS 223
ANEXO E	PRHASE 233
ANEXO F	THE BROWN CORPUS TAG-SET 236
ANEXO G	ABREVIACIONES Y GÉNEROS 248
ANEXO H	CONTRACCIONES 252

RESUMEN

El presente trabajo está enmarcado en el área del Procesamiento del Lenguaje Natural, específicamente la traducción automática. Se justifica en los fenómenos culturales, políticos, económicos y sociales mundiales que implican la necesidad de que los seres humanos tengan acceso a información en idiomas diferentes al nativo. Para suplir esta necesidad por lo menos en un aspecto (acceso a textos en otros idiomas) se plantea la traducción automática como un área de la lingüística computacional en la que se han hecho grandes desarrollos e inversiones desde la segunda mitad del siglo XX hasta el presente. El propósito de la traducción automática es la de traducir por medio de un dispositivo electrónico (computadora) un texto en un lenguaje fuente a su correspondiente representación en el lenguaje destino. Esta labor es demasiado compleja dada la misma complejidad que tienen los lenguajes naturales, así que formalizar un idioma de tal manera que una computadora pueda generar una representación equivalente en otro idioma requiere de muchos elementos y de una gran cantidad de información del mundo. Es así como en la última década se han impulsado mayormente los sistemas de inteligencia artificial, pensando en que si se logra enseñar a una máquina todo el conocimiento del mundo que tiene una persona, será posible entonces que esa máquina pueda generar traducciones tan buenas o mejores que las de un traductor humano. Esto aún sigue siendo un sueño y hasta la actualidad no se conocen de sistemas que estén cerca de este punto.

El principal problema de los sistemas de traducción automática son las ambigüedades que se generan cuando se pasa al lenguaje destino. En la actualidad existen muchos sistemas de traducción automática que permiten generar traducciones de muchos idiomas fuente a muchos idiomas destino y aunque la calidad de las traducciones de algunos son lo suficientemente buenas como para considerarse que pueden ser utilizadas de una forma seria, ha sido imposible evitar que se tengan salidas ininteligibles, pero que para la computadora es difícil determinar si son o no correctas.

El trabajo comienza explorando la historia y estado actual de la traducción automática. En los siguientes capítulos se exploran cada una de las etapas que hacen parte del proceso de traducción automática, documentando la base teórica y generando un prototipo que permita mostrar el procesamiento realizado por cada etapa, permitiendo al lector diferenciar por medio de ejemplos los diferentes procesos que aquí intervienen. Finalmente se muestra la salida del prototipo general para frases afirmativas en presente simple del idioma inglés, teniendo como destino el idioma español, mostrando como se genera una salida libre de ambigüedades gracias al manejo de las relaciones temáticas.

INTRODUCCIÓN

En el desarrollo de este trabajo se abordará el tema de la traducción automática. En primer lugar se hará una documentación del estado del arte de la traducción automática. Se realizará una definición clara y específica de la tecnología llamada traducción automática. Igualmente se definirán cada una de las etapas de dicho proceso. Para cada una de estas etapas se documentarán los problemas asociados a ellas y se plantearán formas de resolverlos. En cada una de las etapas se realizará el desarrollo de un prototipo que permitirá mostrar la funcionalidad de la traducción automática el cual tendrá una explicación detallada y vendrá soportado por ejemplos. Específicamente, este desarrollo se hará usando el método de traducción automática por transferencia, con la particularidad de incluir un análisis semántico.

Se girará en torno a asimilar una base teórica, para así implementar prototipos de cada una de las etapas del proceso de traducción, que permitan al lector comprender y asimilar la funcionalidad de cada uno de los prototipos.

Se pretende documentar el estado del arte desde la historia de la TA hasta las nuevas tecnologías usadas para este proceso, definir claramente en cada etapa el papel que ocupa en el proceso de traducción, además de mostrar ejemplos y casos prácticos que permitan observar la funcionalidad de los prototipos en cada una de las fases.

Este documento se realiza con el fin de recopilar ideas en el campo de la traducción automática, considerando que la traducción es un elemento fundamental para la comunicación de los seres humanos, más aún presentándose fenómenos como el de la globalización que obliga a un permanente intercambio de ideas en diferentes lenguajes. De igual forma, es necesario considerar que el tema de la traducción automática se encuentra dentro de los temas actuales de investigación. Presentándose los desarrollos que en este tema se hacen como tecnología de punta, ya que esta relacionado con uno de los ámbitos más estudiados en estos momentos, el procesamiento del lenguaje natural.

1. FUNDAMENTOS GENERALES

La **traducción** es un proceso que comprende la interpretación del significado de un texto en un idioma (llamado texto origen) a un texto equivalente en otro idioma (llamado texto destino).

Tradicionalmente, la traducción ha sido una actividad desarrollada por humanos, aunque hay numerosos intentos de automatizar la tarea de traducir textos naturales (traducción automática) o de utilizar computadores para ayudar a esta tarea (traducción asistida por computador).

El objetivo de la traducción es crear una relación de equivalencia entre el **texto origen** y el **texto destino**, es decir, garantizar que ambos textos comuniquen el mismo mensaje, a la vez que se tengan en cuenta una serie de limitaciones. Estas limitaciones incluyen el contexto, las reglas de la gramática de cada uno de los idiomas, sus convenciones estilísticas, frases hechas y similares.

La traducción es una disciplina que existe como tal hace poco, pero que como práctica se cree que existe desde los primeros contactos entre humanos de distintas lenguas, como mínimo. Se trata de poner en un código de signos lo que ya estaba en otro código de signos.

1.1. TRADUCCIÓN AUTOMÁTICA

La **Traducción Automática (TA)** proviene del término en inglés **Machine Translation (MT)**, es el proceso de traducción sistematizada de un **lenguaje natural**¹ a otro por medio de un computador. Es una de las investigaciones más antiguas de la ciencia de los computadores, la TA ha probado ser una meta muy difícil de alcanzar, pero hoy, un gran número de sistemas tienen la capacidad de entregar un resultado que si bien no es perfecto, es de suficiente calidad para ser usado en un gran número de áreas y sobre todo para asistir las traducciones humanas.

El concepto de TA en la era de la comunicación y de las nuevas tecnologías busca que la difusión de información y de ideas no tenga obstáculos, por lo que la diversidad de lenguas existentes no debería constituir una barrera.

En las aplicaciones de TA actuales, los niveles de calidad obtenidos para lenguas romances (español, portugués, catalán o gallego) son muy altos. Sin embargo, los

¹ Se conoce por lenguaje natural el idioma con el que se comunican los seres humanos.

resultados empeoran ostensiblemente cuanto más se alejan tipológicamente las lenguas, es decir, mientras más difieran sus rasgos gramaticales, como es el caso de la traducción entre español e inglés. Otro factor muy influyente es el grado de especialización del sistema, atendiendo al tipo de texto y vocabulario que se va a traducir. Por ejemplo, un sistema que se especializa en traducir pronósticos meteorológicos no servirá para la traducción de crónicas deportivas.

Traducir por medios electrónicos es sin duda un desafío científico, pero la razón por la que la traducción automática despierta tanto interés no es de índole científica, sino que se ha convertido en una necesidad práctica.

Al no existir una *lengua franca*², surge con fuerza la necesidad de contar con instrumentos que nos permitan acceder a cualquier tipo de información, independientemente del idioma en que esté escrita o expresada. Es aquí donde la TA se perfila como una de las claves para superar, al menos en parte, los obstáculos en la comunicación.

Es importante decir que los artículos de TA, tienen una calidad inferior a los que están traducidos profesionalmente. Sin embargo, a pesar de los posibles errores lingüísticos y técnicos que hoy en día todavía se producen en el proceso de TA, la calidad de la traducción puede ser suficientemente adecuada para ayudar a resolver los problemas de la comunicación.

*Hubert Murray*³ calculó en 20 millones el número de palabras de información técnica que se generaban en el mundo cada día. Un lector capaz de leer mil palabras por minuto necesitaría 45 días, a una media de ocho horas diarias, para digerir la producción de solo un día. Al cabo de esos 45 días, su desfase sería de cinco años y medio. Una comunidad lingüística necesitaría cientos de traductores para verter a su lengua semejante caudal diario. Según *Susan Hubbard*⁴ comenta en *Information Skills for an Information Society: A Review of Research* que en los últimos 30 años se ha generado más información que en los 5.000 anteriores. Más de 9.000 publicaciones periódicas se editan en Estados Unidos cada año, y casi mil libros salen a la luz diariamente en el mundo.

Cuando Suecia y Finlandia se incorporaron a la Unión Europea en 1995, hubo que traducir alrededor de 60.000 páginas de regulaciones comunitarias⁵. En 1999, gracias al inagotable esfuerzo de los legisladores de Bruselas, esa cifra se había

² Por lengua franca se refiere a una lengua común.

³ MURRAY, Hubert Jr H. M. *Methods for Satisfying the Needs of the Scientist and the Engineer for Scientific and Technical Communication*.

⁴ HUBBARD, Susan H. S. Escritora estadounidense. Profesora de inglés asociada de la Universidad central de Florida.

⁵ Lo que se conoce como el "acquis communautaire"

incrementado en 20.000. La inminente entrada en la Unión Europea de nuevos países del este europeo pone a la **CE** (Comunidad Europea) ante una situación lingüística muy delicada, que sólo los avances de las nuevas tecnologías podrían sobrellevar⁶.

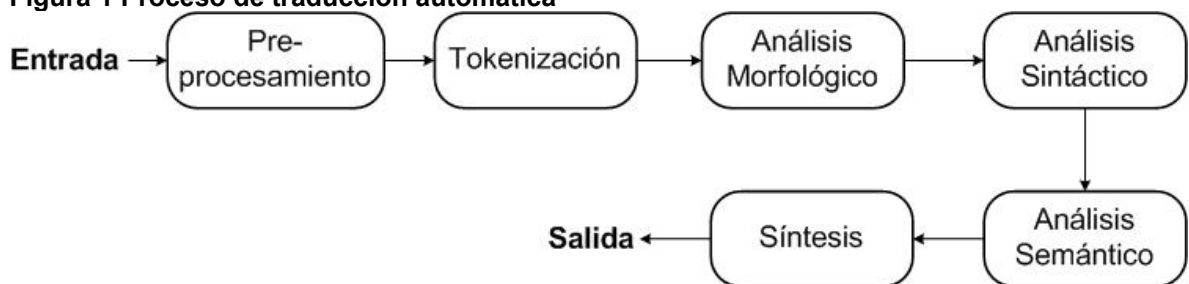
1.2. EL PROCESO DE TRADUCCIÓN

El proceso traductológico contiene las siguientes actividades:

- Decodificar el sentido del texto origen.
- Recodificar este sentido en la lengua meta.

Para realizar este proceso se deben aplicar una serie de procesos, como se ven en la Figura 1.

Figura 1 Proceso de traducción automática



Para decodificar el sentido de un texto, el traductor tiene que identificar en primer lugar las **unidades de traducción** que lo componen, es decir, los segmentos del texto que deben tratarse como una **unidad cognitiva**. Una unidad de traducción puede estar compuesta por una **palabra**, **frase** o incluso una o más **oraciones**.

Tras este procedimiento, sencillo a primera vista, se esconde una operación cognitiva compleja. Para decodificar el sentido completo del texto origen, el traductor tiene que interpretar y analizar todas sus características de forma consciente y metódica. Este proceso requiere un conocimiento profundo de la gramática, semántica, sintaxis y frases hechas o similares de la lengua origen, así como de la cultura de sus hablantes.

⁶ ABAITUA, JOSEBA J. A. "Traducción automática: Presente y Futuro: Cuello de botella de la sociedad de la información". 1 Jun 2005. <http://www.foreignword.com/es/Technology/art/Abaitua/Abaitua_1.htm>.

El traductor debe contar también con estos conocimientos para recodificar el sentido en la lengua meta. De hecho, estos suelen ser más importantes y, por tanto, más profundos que los de la lengua origen. De ahí que la mayoría de los traductores traduzcan a su lengua materna. Además, es esencial que los traductores conozcan el área que se está tratando.

1.3. ETAPAS DEL PROCESO DE TRADUCCIÓN

1.3.1. Preprocesamiento

El **preprocesamiento** tiene como objetivo darle un formato al texto antes de ser sometido a los diferentes módulos de traducción. Este formato consiste en la eliminación de caracteres inútiles y adecuación del texto para hacer más fácil su tratamiento en las fases posteriores.

Esta etapa tiene dos módulos principales: el primero es el **filtro** que destruye los espacios en blanco innecesarios, se enlazan las palabras que han sido separadas por un guión y un salto de línea y como segundo módulo la disolución de **contracciones** que consiste en separar una contracción en sus diferentes componentes, solo en el caso en que las contracciones no generen ambigüedades, utilizando para ello un diccionario externo que especifica cómo se deben descomponer dichas contracciones, de manera que sólo es necesario modificar esta información para adaptar este módulo a otro idioma.

1.3.2. Tokenización

El proceso de **tokenización** es el segundo paso realizado en un proceso de TA, consiste en la identificación de cada una de las piezas que conforman un texto. No se puede simplemente decir que una palabra es una cadena de caracteres con un espacio antes y después, sin duda el tratamiento es un poco más complejo, se deben tener en cuenta casos especiales como: las combinaciones de letras y números en una sola palabra, los caracteres especiales, los signos de puntuación, el formato de la fecha y la hora, las abreviaciones, entre otros.

Para darle solución al proceso de tokenización se emplearán expresiones regulares las cuales pueden explorar un texto y separar las palabras, indicando los caracteres que pueden ser incluidos para que las palabras puedan ser reconocidas.

Así, el texto queda preparado para su posterior etapa consistente en el análisis morfológico

1.3.3. Análisis Morfológico

El **análisis morfológico** es el proceso siguiente a la tokenización y permite identificar los lexemas y morfemas presentes en los tokens, es decir, cada palabra debe ser analizada para identificar las palabras derivadas. Los diccionarios usados en los sistemas de TA no tienen el significado de palabras derivadas de palabras más simples. Las palabras derivadas deben ser identificadas por el programa. Las formas verbales y los plurales son las palabras derivadas más comunes.

Además el **diccionario** le brinda al analizador morfológico la información gramatical adecuada para etiquetar los tokens y prepararlos para la siguiente etapa del proceso de traducción.

Es importante destacar que a partir de este análisis en adelante es de mucha importancia el diccionario, ya que suministra la mayor parte de la información para realizar el proceso de traducción, como categorías gramaticales, géneros, roles semánticos y significados asociados a las palabras.

1.3.4. Análisis Sintáctico

El **análisis sintáctico** es el proceso que se debe realizar después del análisis morfológico, se hablará de los dos enfoques usados para determinar las secuencias gramaticales de los idiomas y se hará uso de uno de estos enfoques para definir formalmente la gramática del idioma inglés para frases afirmativas en presente simple.

El módulo desarrollado en el presente trabajo está apoyado principalmente en un análisis descendente, que consiste de analizar desde lo más general hasta lo más específico, es decir desde el análisis de la estructura de la oración hasta llegar a las unidades léxicas formadas por palabras.

1.3.5. Análisis Semántico

Después de realizar el análisis sintáctico pueden permanecer algunas ambigüedades. La información semántica puede ser usada para resolver el problema. Aquí es donde se observa el por qué un buen diccionario debería tener información semántica de las palabras. Por ejemplo, las palabras relacionadas con música deberían estar marcadas como tal con una etiqueta semántica.

El análisis semántico es una de las etapas más importantes, ya que es el encargado de tomar decisiones para que las frases que sintácticamente son correctas sean así mismo válidas a la luz de la semántica, ya que en este punto dichas frases pueden tener diferente interpretación de acuerdo al contexto. Es

importante recalcar que no es muy común observar que los traductores empleen información semántica para realizar el proceso de traducción.

1.3.6. Síntesis al lenguaje destino

La síntesis consiste en trasladar la estructura sintáctica y semántica que se tiene al lenguaje destino, para ello es necesario un conjugador de verbos y un generador de plurales, así como un análisis de géneros, número y persona gramatical.

La síntesis es la última etapa en el proceso de traducción automática, aunque algunos autores incluyen una etapa de post edición en la cual se refina el documento obtenido. Puede decirse que el proceso de traducción automática termina en este punto.

2. ESTADO DEL ARTE

2.1. LA NECESIDAD DE LA TRADUCCIÓN

2.1.1. El problema: Diversidad de idiomas

En la actualidad existen 6.912 idiomas en todo el mundo⁷, de los cuales el chino, el español, el inglés, el hindú y el árabe son en la actualidad los idiomas más hablados en el mundo con alrededor de 2.000 millones de personas. En la Tabla 1 se muestra un cuadro comparativo con estimaciones de los idiomas más hablados según datos de diferentes fuentes:

Tabla 1 Clasificación de idiomas por número de parlantes

Lenguaje	ETH	ALM	CIA	SIL	ENC	WEB	Promedio
Chino, Mandarín	885.0	874	874		836	1200	874.00
Español	332.0	322	339	332	332	332	332.00
Inglés	322.0	309	341	322	322	322	322.00
Hindi	182.0	496	366	182	333	250	291.50
Árabe		256		174	186	200	193.00
Bengali	189.0	215	207	189	189	185	189.00
Indonesio		176					176.00
Portugués	181.0	194	167	170	170	160	170.00
Ruso	145.0	275	160	170	170	160	165.00
Japonés	127.0		125	125	125	125	125.00
Alemán	120.0		100	98	98	100	100.00
Chino, Wu	92.5						92.50
Punjabi	89.0					90	89.50
Javanés	75.5					80	77.75
Coreano	75.0		78				76.50
Francés	72.0	129	77	79	72	75	76.00
Marathi	71.8						71.80

Los valores están representados en millones y al final se muestra el promedio de parlantes según los diferentes datos.

- ETH = Ethnologue (1999). <http://www.ethnologue.com/>

⁷ "Ethnologue" 1 Jun 2005. <<http://www.ethnologue.com>>.

- ALM = Almanaque Mundial (2005). <http://www.worldalmanac.com>
- CIA = CIA Factbook (2000). <http://www.cia.gov/cia/publications/factbook/>

Las dificultades que acarrearán las divergencias del lenguaje se ven claramente en las comunidades donde se habla más de un idioma, la solución de adoptar una lengua en común no es una alternativa muy atractiva dado que se deben tener en cuenta factores políticos y sociales, como la dominación del idioma seleccionado y las desventajas para los que hablan otros idiomas, teniendo en cuenta además que la desaparición de un lenguaje involucra también la desaparición de la cultura que lo distingue y su manera de pensar, esta pérdida por lo tanto debería importarle a todo el mundo.

2.1.2. La solución: Traducción

La **traducción** de idiomas es, en su definición más básica, la transformación de texto de un idioma a otro. La traducción es necesaria para la comunicación, para la interacción humana ordinaria, para obtener la información que se necesita para formar parte de la sociedad⁸. El problema es que la demanda de traducciones en el mundo moderno es mucho mayor que lo que se puede proveer. Parte del problema radica en que hay muy pocas personas dedicadas a traducir y en que hay limitantes en la productividad que estos pueden alcanzar⁹.

Para traducir se requiere examinar un texto original escrito en lo que se conoce como **lenguaje fuente** y escribir un texto equivalente en otro idioma, llamado **lenguaje destino**, con el objetivo de mantener el tono y el significado del texto original. Traducir no es sencillo. No se trata de sustituir una palabra por otra, sino de poder reconocer todas las palabras de una frase y la influencia que tienen las unas sobre las otras. Esta relación de influencia es la que produce ambigüedades, pues las palabras tienen un significado según el contexto donde se encuentren. Hasta el texto más simple puede estar plagado de ambigüedades.

La profesión de traductor data de muchos siglos atrás. Sin embargo, el rol actual del traductor es muy diferente al de aquellos días. Ahora no sólo se necesita la pericia de los traductores, sino que también es necesaria la intervención de la tecnología dentro de este proceso. El contenido ha evolucionado del papel a la electrónica y así, los dispositivos de asistencia computarizada se han convertido en una herramienta comúnmente utilizada en la traducción de texto. Los

⁸ ARNOLD, Doug D. A. "Machine Translation: an Introductory Guide" NCC BlackWell (1994)

⁹ Ibid.

traductores distinguen la traducción de la *interpretación*, que consiste en la traducción oral de un discurso.

La traducción es necesaria para la comunicación¹⁰, ya que la recolección de información debería darse en la totalidad de la sociedad y esto resulta difícil por la cantidad de idiomas que existen en el mundo, para las personas en general es complicado dominarlos a todos de una manera integral, restringiéndoles expresar lo investigado en su idioma natural y recibir información que afecta directamente el medio donde se encuentran. El problema es que al traducir generalmente se cambia el sentido de lo que se quería expresar y es ahí donde se encuentra la habilidad para traducir.

En la actualidad la demanda de traducciones en el mundo moderno es mucho mayor que lo que se puede llegar a suplir, parte del problema es que hay pocas personas dedicadas a traducir y su habilidad para traducir se ve limitada. Ahora bien, la automatización puede ayudar a incrementar la productividad al traducir, siendo entonces la automatización de este proceso una necesidad social y política para las sociedades modernas en el sentido que no es tan sencillo imponer un idioma en común para los miembros que la componen.

En el aspecto económico la traducción por si misma es de gran importancia. Por ejemplo, es claro que si un comprador tuviera que escoger un producto con los manuales en español y otro con sus manuales en japonés, la mayoría de hispano-parlantes se inclinarán por el primero de ellos o por lo menos el idioma en que éste se presente representaría un fuerte criterio de selección. Ahora bien un fabricante desearía que su producto cuyos manuales fueron escritos inicialmente en un idioma sean traducidos a otros idiomas para poder atacar mercados con personas que lo hablen.

Debe tenerse en cuenta también que la traducción es una actividad costosa, requiere mucho más que únicamente conocer un gran número de idiomas, y en algunos países la profesión de traductor es casi tan bien remunerada como la de profesionales de mucha experiencia. Más aún los retardos en la traducción representan grandes pérdidas. Los estimativos varían, pero al producir textos traducidos con alta calidad de un material complicado un traductor profesional puede promediar no más de cuatro a seis páginas por día y esto puede ser un factor para que se produzcan retardos al traducir la documentación de un producto el cual será lanzado al mercado en poco tiempo. Ha sido estimado que cerca del 40 al 45 por ciento de los costos de las instituciones de la comunidad europea se deben a costos por idiomas, de los cuales la traducción e interpretación son los

¹⁰ Ibid.

principales elementos, esto puede representar cerca de 300 millones de Euros por año¹¹.

2.2. EL SECTOR DE LA TRADUCCIÓN

Un número de diversas profesiones contribuyen al sector de la traducción de la industria del lenguaje, y éstas se describen abajo. Cuando es relevante, se hace una breve mención de la tecnología de idiomas usada en estas profesiones.

2.2.1. Traducción

Los traductores son los especialistas de la comunicación que trabajan con material escrito. Ellos se esfuerzan para transferir el contenido de un texto de un lenguaje a otro de una manera que tal que pueda reflejar el mensaje verdadero como el estilo del autor original.

La decisión acerca de si utilizar o no herramientas (y si es así, qué clase de herramientas) para ayudarse con el proceso de traducción depende mucho del tipo de texto que es traducido.

2.2.2. Revisión

Los revisores son típicamente traductores de experiencia que se encargan de verificar y, si es necesario, corregir las traducciones producidas por los traductores con menos experiencia. A menudo, los trabajos se anuncian con el título *Traductor/Revisor*, que significa que el candidato pasará parte de su tiempo haciendo trabajo de traducción y el resto revisando traducciones producidas por otros.

2.2.3. Interpretación

Los intérpretes son los profesionales de la comunicación que trabajan con el lenguaje hablado. Su trabajo es transportar exactamente el contenido y el tono de los mensajes de un lenguaje a otro.

A diferencia de los traductores, que típicamente tienen la oportunidad de hacer alteraciones y mejoras a sus textos antes de entregar una versión final, los

¹¹ PATTERSON, Wright W. P. Civil Engineering Center, Estimaciones de costos de traducciones. 1992.

intérpretes tienen que hacer su trabajo en tiempo real¹², sin la oportunidad de retractarse y hacer revisiones. Por lo tanto, los intérpretes deben asegurarse de que conocen la información base que necesiten, ya que generalmente no es posible consultar textos de referencia durante el proceso de interpretación. Entonces, mientras que los intérpretes pueden utilizar tecnología de traducción para ayudarse con su investigación base, no les es posible utilizar tales herramientas mientras que están interpretando. Se puede distinguir entre los diferentes tipos de interpretación, bien sea por el contexto en el que ocurren (por ejemplo: *interpretación en conferencia*, *interpretación en la corte*), o por la manera en que se realizan (por ejemplo: *interpretación simultánea*, *interpretación consecutiva*, *interpretación bilateral*).

2.2.4. Terminología

Los terminólogos son especialistas de la investigación que estudian el vocabulario (y los conceptos señalados por este vocabulario) de dominios concretos. Entre las tareas que típicamente realizan estas personas están la de identificar y definir términos/conceptos particulares a un dominio en específico y a compilar estos datos de una forma que puedan ser fácilmente utilizados y/o impresos como un glosario. Cuando los terminólogos trabajan en entornos multilingües, procuran establecer los equivalentes de términos en otros idiomas. Los terminólogos también pueden trabajar en el área de estandarización, donde procuran establecer listas de términos preferidos.

Con respecto a las herramientas tecnológicas para idiomas, *Pavel y Nolet*¹³ precisan que aunque cualquier actividad terminológica se puede realizar manualmente, la automatización permite mejoras sin precedentes en productividad, calidad y accesibilidad. Como resultado, el uso de la tecnología en la disciplina de la terminología está en apogeo.

2.2.5. Localización

La localización es el tipo de traducción que debe tener en cuenta los aspectos sociales, científicos y culturales de una región.

Los especialistas de la localización, llamados a veces como tecnolingüistas, son relativamente nuevos profesionales de la lengua. Se especializan en traducir

¹² Para este caso tiempo real significa simultáneamente.

¹³ PAVEL, Silvia S. P. and NOLET, Diane D. N. 2002. MANUAL DE TERMINOLOGÍA. Ministerio de Obras Públicas y Servicios Gubernamentales de Canadá.

medios electrónicos como sitios **Web**¹⁴ y *software*. La localización también puede ser extendida a traducir la documentación, materiales de publicidad, etc. Esencialmente, la localización implica el adaptar el material para cumplir los requerimientos lingüísticos, culturales y técnicos del nicho de mercado destino.

La localización es más que la simple traducción de textos, este trabajo puede implicar el localizar características tales como fechas, épocas, modernidades, imágenes e incluso colores. Los especialistas de la localización también deben enfrentar desafíos que no se encuentran típicamente en otros tipos de traducción. Por ejemplo, pueden estar restringidos por la cantidad de espacio disponible en la pantalla de la computadora y sus componentes (por ejemplo: *botones*, *menús*), pueden tener que traducir segmentos de texto en el cual diversas variables se pueden sustituir diferentes veces y posiblemente tengan que reasignar teclas rápidas (por ejemplo: *Ctrl-p* para la impresión o en inglés *print* puede convertirse en *Ctrl-i* para *imprimir*).

Dado que estos profesionales trabajan con medios electrónicos, típicamente tienen un conocimiento profundo de las herramientas tecnológicas para idiomas. A diferencia de los traductores, intérpretes terminólogos, quienes pueden trabajar individualmente o con colegas de la lengua, los especialistas de la localización generalmente forman parte de un equipo multidisciplinario que también incluye profesionales como gerentes de proyectos, ingenieros de software, coordinadores de calidad, diseñadores gráficos, etc.

2.2.6. Edición

Los editores son los profesionales que revisan la salida de los sistemas de TA y la editan para que pueda ser entregada a un cliente. Dependiendo de las necesidades del mismo, la edición puede ser mínima, por ejemplo si es importante que los lectores entiendan el significado del texto, pero el estilo es menos importante, o bien puede ser completo, por ejemplo si el texto pretende ser diseminado y por lo tanto necesita ser exacto y legible. A medida que aumenta el número de compañías que utiliza la TA, probablemente habrá más demanda de trabajo para los editores. Entre las organizaciones que han anunciado recientemente posibilidades para los editores están: Desarrollo de Recursos Humanos de Canadá, SDL Internacional¹⁵ y la Organización Panamericana de la Salud.

¹⁴ La Web o WWW (World Wide Web), en español: Telaraña mundial

¹⁵ "SDL International". 1 Jun 2005. <<http://www.sdl.com>>. Es el mayor proveedor del mundo de soluciones tecnológicas para la gestión global de información.

2.2.7. Traducción Audiovisual

La traducción audiovisual, también conocida como traducción de pantalla, consiste en dos técnicas principales: doblaje y el subtitular. Se puede referir al doblaje en un sentido amplio o estrecho. En su término más amplio, el doblaje se refiere a cualquier técnica para cubrir la voz original de una producción audiovisual con otra voz, que puede incluir técnicas de reexpresión, tales como narración o comentarios libres. En su definición mas estricta, el doblaje se refiere mas específicamente a aquel de labios sincronizados, donde el dialogo traducido se ajusta a los movimientos de la boca del actor para dar la impresión de que el actor a quien la audiencia ve, esta hablando realmente en el idioma destino. El doblaje es un proceso que implica muchas etapas, además de la transferencia del idioma, es por eso que allí encontramos factores adicionales (por ejemplo equipos, la escogencia de actores, la habilidad del editor, estándares de ingeniería de sonido) que contribuyen a la calidad del doblaje. Además, quienes hacen doblaje (en inglés *dubbers*) enfrentan retos que no hacen parte de otros tipos de traducción más convencionales. Por ejemplo, la restricción de sincronización puede obligarlos a introducir cambios importantes al dialogo original simplemente para emparejar los sonidos a los movimientos del labio y a otros gestos físicos.

Subtitular se refiere al proceso de proporcionar subtítulos sincronizados en el idioma destino para producciones como películas, televisión e incluso opera en vivo. Al igual que los *dubbers*, quienes hacen subtítulos (en inglés *subtitlers*) trabajan bajo ciertas restricciones adicionales a las que tienen que ver con otro tipo de traducción. Por ejemplo, los subtítulos se limitan generalmente a dos líneas con un máximo de 35 caracteres cada una, de modo que el texto no cubra demasiado de la imagen y para poder sincronizar el subtitulo con el dialogo. Esto significa que hay limitaciones serias en la cantidad de información que puede ser transmitida en la mayoría de los subtítulos. Por lo tanto, el subtitular exige generalmente una compresión total del material original, así en algunas ocasiones el *subtitled* puede necesitar agregar información adicional, tal como una explicación de una referencia cultural, que de otra manera podría ser confusa para la audiencia.

2.3. ANÁLISIS DEL SECTOR DE LA TA

El ambiente en el que actualmente se desenvuelve la TA, es un ambiente que se ha visto ligeramente limitado por las inconsistencias presentadas al realizarse las traducciones y el tiempo que se requiere para realizar un traducción satisfactoria.

Las empresas del medio se ven avocadas a realizar grandes inversiones en maquinaria para automatizar sus procesos y para agilizar sus comunicaciones, lastimosamente en nuestro país no se cuenta con la tecnología para investigar y

desarrollar maquinaria, esto obliga a las empresas a realizar importaciones; y es allí donde se dispara la inversión pues no se cuenta con el personal capacitado para la instalación y manejo de la maquinaria.

Pero hay algo que no tiene variación, hablese de lenguas, países o razas y es el conocimiento, la formación base no tiene muchas variaciones de una lengua a otra. Un profesional bien formado debe estar en la capacidad de interpretar un manual de instalación y funciones de una maquinada determinada que se encuentre desarrollada dentro de su área de conocimiento.

Debido a la inherente globalización y alta competencia que se presenta en nuestra sociedad, se genera una necesidad para las diferentes empresas la cual radica en la apropiación de un esquema de traducción que facilite la comunicación y las diferentes formas de expresión, ya que es necesario intercambiar información con el extranjero, y apropiarse de nuevas tecnologías e información, para estar a la vanguardia de la tecnología y no permitir ser absorbido; pues esto es causa de estancamiento y posible extinción en un futuro.

Lograr obtener la precisión de la traducción humana, sería el reto a alcanzar con un sistema de TA, pero lograr obtener borradores que aproximen al resultado esperado en muy poco tiempo si sería un gran avance.

Este ocuparía un nuevo espacio de la torta de mercado en la que los sistemas automatizados podrían entrar a ocupar una porción más alentadora, estos son relegados en su mayoría por sus inconsistencias e interpretaciones erradas y confusas, lo cual no ocurre cuando la traducción es realizada por un profesional aplicando diferentes conceptos lingüísticos.

En la actualidad se estiman alrededor de 500 sistemas automatizados de traducción en venta¹⁶. La calidad no es tan buena como se decía anteriormente comparada con la de un traductor profesional pero los resultados son buenos para usuarios caseros. La precisión que un sistema automatizado de traducción debe tener para aplicaciones empresariales debe ser muy confiable, para permitir interpretaciones claras, ya que en la mayoría de los casos se requiere de terminología técnica la cual de ser alterada generaría confusiones severas. En este aspecto es en el que los traductores profesionales llevan la mayor ventaja ya que utilizan una variedad de procesos, habilidades y de recursos del pensamiento para interpretar el significado de una oración y para comunicar el significado de

¹⁶ HUTCHINS, John J. H., HARTMANN, Walter W. H., and ITO, Etsuo E. I. EUROPEAN ASSOCIATION FOR MACHINE TRANSLATION, Compendium of Translation Software. 2005.

esa oración en una diversa lengua. Son expertos en la gramática apropiada, el contexto de la frase y el vocabulario adjunto a la frase¹⁷.

2.4. DIVISIÓN DE LOS SISTEMAS DE TA

Los sistemas de TA pueden ser clasificados por el número de idiomas que soportan, por la dirección de la traducción y por el grado de automatización así:

2.4.1. Cantidad de idiomas soportados

Se dividen en sistemas **bilingües** o **multilingües**. Los **sistemas bilingües** están desarrollados para un único par de lenguas. En cambio, los **sistemas multilingües** se desarrollaron para traducir con diversos pares de lenguas: de una o varias lenguas origen son capaces de traducir a una o varias lenguas destino.

2.4.2. Dirección de la traducción

Las traducciones pueden hacerse en una única dirección, por ejemplo, del inglés al español; en este caso la traducción sería **unidireccional**. Son **bidireccionales** aquellos que traducen de una lengua a otra y viceversa, por ejemplo del inglés al español y del español al inglés.

2.4.3. Grado de Automatización

Los sistemas de **traducción totalmente automática**, son aquellos que realizan la traducción sin intervención alguna del traductor humano (salvo en la revisión).

En los sistemas de **traducción asistida** la intervención del traductor humano es constante.

En los sistemas de **traducción humana asistida por computador**, el computador se utiliza como herramienta de ayuda al traductor, facilitándole el uso de, por ejemplo, procesadores de textos con diccionarios en línea, información gramatical y morfológica de las palabras, verificación ortográfica, etc.

¹⁷ "Automated Real-Time Translation". 2001. 1 Jun 2005. <<http://www.sdl.com/es/localization-information/white-papers-articles/white-papers-list/white-papers-automated-real-time-translation.htm>>.

2.5. USO DE SISTEMAS DE TA

De acuerdo a la calidad de traducción necesitada los diversos tipos de sistemas de TA pueden ser clasificados de la siguiente manera:

2.5.1. Diseminación

La producción de traducciones de calidad publicable, no necesariamente textos que en realidad vayan a ser publicados, pero sí de tal calidad. Este tipo de textos son requeridos usualmente por compañías y necesitan de traductores profesionales. La salida normal de un sistema de TA es de calidad baja. Calidad publicable significa asistencia humana, revisando la salida del texto, pre-editando la entrada, usando un lenguaje controlado o restringiendo a un dominio o contexto específico. Se ha encontrado (a partir de la experiencia usando sistemas de TA) que mientras más se restrinja el dominio de aplicación del idioma, mejor es la calidad de la traducción.

2.5.2. Asimilación

Traducción de textos para filtrar información, o la traducción de textos para usuarios ocasionales, esto es, público en general. La salida de estos sistemas no necesita ser editada ya que se puede aceptar la baja calidad, pues el propósito es solamente el de obtener una idea del texto. De hecho, muchos de los sistemas de TA son útiles solo para esta función, ya que sus salidas no son satisfactorias como borradores para tareas de diseminación.

2.5.3. Intercambio

La comunicación en diferentes idiomas entre individuos, por correspondencia, correo electrónico o telefónicamente. Aquí igualmente la calidad de la traducción no es tan importante, mientras las personas obtengan la información que necesitan, entiendan el mensaje que reciben o intenten indicar sus intenciones.

2.5.4. Acceso a Bases de Datos (BD)

El uso de la traducción como herramienta para obtener información de una BD en un idioma extranjero, no muy conocido por el usuario. Esto es lo mismo que decir: el uso de ayudas de traducción para buscar principalmente en Internet, para acceder a páginas *Web*.

Mientras la diseminación y asimilación han sido usos tradicionales de la TA y herramientas de traducción desde el principio, su uso para el intercambio y el

acceso a BD son aplicaciones más recientes que están creciendo rápidamente, esto debido al crecimiento de Internet.

2.6. ENFOQUES DE LA TA

2.6.1. Enfoques tradicionales a la TA

Desde el punto de vista del diseño hay tres enfoques diferentes de TA. Los sistemas adoptan un determinado tipo de diseño que permite que las traducciones se desarrollen de una forma u otra, de tal forma que varía el resultado obtenido. Los diferentes sistemas son: los sistemas directos, los de transferencia y los sistemas interlingua.

2.6.1.1. Los sistemas de traducción directa

Podrían compararse con grandes diccionarios. Generalmente realizan la traducción casi palabra por palabra, ya que la información sintáctica que poseen es mínima. Por ello los resultados que ofrecen suelen ser bastante pobres.

2.6.1.2. Los sistemas de transferencia

Contienen además de grandes léxicos bilingües y multilingües, un amplio conocimiento sintáctico y semántico de las lenguas tratadas. Esto permite traducir palabras de una lengua a otra teniendo en cuenta además el contexto morfológico, sintáctico y semántico de la frase. Es capaz de llevar a cabo también la transferencia estructural, es decir, los cambios en el orden de elementos y en la estructura de la frase para adecuarse a la de cada lengua.

2.6.1.3. Los sistemas interlingua

Se lleva a cabo un análisis mucho más profundo de la frase. La idea detrás de este enfoque, es el crear un lenguaje artificial conocido como *interlingua*, el cual comparte todas las características y hace todas las distinciones entre todos los idiomas. Para traducir entre dos idiomas diferentes, se usa un analizador para convertir el texto del idioma fuente al interlingua, y un generador que convierte el interlingua en el texto en el idioma objetivo. Aunque teóricamente se trataría del mejor enfoque de los tres, en realidad estos sistemas están en fase de laboratorio o se utilizan para aplicaciones muy restringidas, debido a los problemas prácticos que presentan en el diseño y la implementación de una interlingua eficaz.

Hay que tener en cuenta que no existen sistemas puros de traducción directa, de transferencia o interlingua, sino sistemas que se aproximan más a un enfoque determinado, pero que también pueden tener características de los otros sistemas.

2.6.2. Enfoques modernos de TA

Además de los enfoques explicados anteriormente existen otros enfoques que se han dedicado a estudiar la TA, especialmente lo que concierne a la **TA basada en el conocimiento**. Aunque *Hutchins*¹⁸ en su libro no hace mención a estos enfoques, actualmente hay numerosos estudios teóricos y proyectos que abandonaron la línea tradicional y se han ocupado en el estudio de los enfoques más novedosos al problema de traducir mediante computadoras. En realidad, las bases de estos enfoques estaban ya sentadas desde mucho tiempo atrás, pero ha sido en los últimos años cuando se han llevado a la práctica en sistemas reales. Muy genéricamente cabría distinguir dos tipos radicalmente diferentes de paradigmas:

- Empíricos
- Simbólicos (o basados en reglas)

Los segundos continúan la tradición de los sistemas anteriores en el sentido de que se basan en la utilización de información/conocimiento (ya sea lingüística o del mundo) y una serie de reglas que explicitan cómo debe ser utilizado dicho conocimiento. De este modo, la TA basada en el conocimiento se puede considerar como una continuación de algunos de los tipos de metodologías tradicionalmente usadas (interlingua), aunque los nuevos sistemas mantienen grandes diferencias con los primitivos. Estas diferencias (por ejemplo, la utilización de grandes bases de conocimiento) están motivadas especialmente por la experiencia acumulada.

Los enfoques empíricos, en cambio, suponen una ruptura radical con este tipo de metodologías (tanto de las de interlingua como de las de transferencia). Aprovechando la disponibilidad de grandes cantidades de texto en formato magnético, se ha intentado aplicar técnicas estadísticas o de búsqueda de patrones, tanto para traducir como para extraer información para ser utilizada posteriormente en el proceso de traducción. Dentro de estas nuevas metodologías

¹⁸ HUTCHINS, John J. H. Lingüista inglés especializado en traducción automática. Publicaciones: "Early years in machine translation: memoirs and biographies of pioneers", "An introduction to machine translation", "Machine translation: past, present, future", entre otras.

cabe distinguir dos enfoques distintos: la TA basada en estadística y la TA basada en el ejemplo.

2.6.3. Representación del conocimiento

En lo que respecta a la representación del conocimiento, se distinguen principalmente dos tipos de esquemas de representación para la creación de bases de conocimiento:

2.6.3.1. Esquemas lógicos

La lógica fue uno de los primeros formalismos usados por los investigadores de inteligencia artificial para representar estructuras de conocimiento en computadores digitales. El propósito de usar la lógica como medio de representación de conocimiento es aportar una notación que sea lo suficientemente versátil para cubrir los conceptos deseados y al mismo tiempo sea capaz de soportar procesos deductivos¹⁹.

Un formalismo de representación basado en la lógica permite expresar muchos tipos de generalizaciones, incluso sin disponer de una descripción completa de la situación. El uso de la deducción también permite efectuar consultas lógicamente complejas a una base de conocimiento que contenga dichas generalizaciones, incluso cuando una consulta no pueda ser evaluada directamente (*Cohen & Feigenbaum 1982*²⁰).

2.6.3.2. Esquemas de redes semánticas

Los esquemas de redes semánticas tienen una fundamentación psicológica muy sólida, por lo que se han realizado numerosos esfuerzos por llevar a cabo implementaciones importantes basadas en ellas.

Las redes semánticas han sido muy utilizadas en inteligencia artificial para representar el conocimiento y por tanto ha existido una gran diversificación de técnicas. Los elementos básicos que se encuentran en todos los esquemas de redes son:

- Estructuras de datos en nodos, que representan conceptos, unidas por arcos que representan las relaciones entre los conceptos.

¹⁹ "Esquemas Lógicos". 1 Jun 2005. <<http://elies.rediris.es/elies9/4-3-1.htm>>.

²⁰ Ibid.

- Un conjunto de procedimientos de inferencia que operan sobre las estructuras de datos.

Básicamente, se pueden distinguir tres categorías de redes semánticas:

- **Redes IS-A:** En las que los enlaces entre nodos están etiquetados.
- **Grafos conceptuales:** En los que existen dos tipos de nodos: de conceptos y de relaciones.
- **Redes de marcos:** En los que los puntos de unión de los enlaces son parte de la etiqueta del nodo.

En general, cuando se habla de redes semánticas se suele hacer referencia a uno de estos esquemas, normalmente a las redes IS-A o a los esquemas basados en marcos, que comparten ciertas características fundamentales. Entre estas características compartidas se destaca la herencia por defecto (*default inheritance* en inglés). En una red semántica, los conceptos (o estructuras, clases, marcos, dependiendo del esquema concreto) están organizados en una red en la que existe un nodo superior (*top*) al que se le asigna uno o varios nodos hijos, que a su vez tienen otros conceptos hijos y así sucesivamente hasta que se alcanza el final (*bottom*), cuyos nodos ya no son conceptos sino instancias.

Los esquemas de representación lógicos, durante mucho tiempo fueron considerados como la respuesta a todas las necesidades de los sistemas artificiales de comprensión del lenguaje natural, debido principalmente a la enorme repercusión del trabajo del gran lógico y matemático *Richard Montague*²¹. En la actualidad, sin embargo, son pocos los enfoques que pretenden utilizar la lógica como único recurso, aunque ésta se encuentra embebida en la mayoría de los sistemas de representación y *procesamiento del lenguaje natural PLN*²².

Los defensores de los enfoques basados en conocimiento justifican el uso de estas técnicas basadas en las siguientes razones:

- La traducción de ***lenguas naturales*** es un proceso complejo que requiere una vasta cantidad de información léxica y de conocimiento del mundo.

²¹ MONTAGUE, Richard R. H. (California, 1930 - Los Angeles 1971). Lógico y filósofo.

²² "Bases de datos y bases de conocimiento". 1 Jun 2005. <<http://elies.rediris.es/elies18/522.html>>.

- Los enfoques no lingüísticos tropiezan con obstáculos que son fácilmente salvables mediante técnicas lingüísticas.
- Los sistemas de representación del conocimiento actuales permiten una utilización del conocimiento humano para muchas tareas complejas, entre ellas la traducción.
- Tratar de reproducir de forma artificial las estructuras de conocimiento que un traductor humano emplea puede llevar, cuando menos, a un mejor entendimiento de estas estructuras de conocimiento y de los procesos que tienen lugar en la mente humana en lo que respecta a la comprensión del lenguaje.
- Tener como objetivo la creación de un sistema de **KBMT**²³ implica, en cualquier caso, la elaboración de un vasto repositorio de información lingüística y de conocimiento del mundo. Si empleamos las técnicas adecuadas, esta información podrá ser reutilizable en otras muchas tareas de PLN.
- La experiencia acumulada en el campo de la TA apunta en la dirección de estos sistemas, que suponen una inversión segura a largo plazo.

Como conclusión general en cuanto a los enfoques, los expertos en el tema de la TA recomiendan que no existe uno definitivo que resuelva todos los problemas de la TA. Los sistemas con más éxito son aquéllos que adoptan una postura ecléctica, integrando técnicas de diversos enfoques.

2.7. HISTORIA DE LA TA

La TA como punto de interés para los investigadores (lingüistas, ingenieros, entre otros) se presenta en un continuo estudio que se inicia con *Leibnitz*²⁴ en el siglo XVII, que propone el uso de dispositivos mecánicos para salvar la barrera del idioma, en ese caso el latín. Su propuesta fue la de utilizar un código numérico para mediar entre las lenguas vernáculas y el latín como lengua universal de la ciencia. En esa época otra propuesta para vencer la barrera del idioma fue hecha por *Descartes*²⁵, que proponía un lenguaje universal a través de un código para cada uno de ellos. Esta propuesta es quizá el antecedente más remoto de los

²³ KBMT (Knowledge-Based Machine Translation) En español: Traducción automática basada en el conocimiento.

²⁴ LEIBNIZ, Gottfried G. L. Filósofo racionalista alemán (Leipzig, 1646 - Hannover, 1716).

²⁵ DESCARTES, Rene R. D. Filósofo y matemático francés (La Haye, Turena, 1596 - Estocolmo, 1650).

primeros intentos de TA del presente siglo, a través de la construcción de diccionarios mecánicos.

En 1668 cuando *Wilkins*²⁶ publicó su obra *An Essay towards a real character and Philosophical Language*, donde expone sus ideas acerca de las bases lógico-rationales para el establecimiento de equivalencias interlingüísticas, bases teóricas bajo las cuales se inician los trabajos de clasificación universal de los conceptos y entidades. A lo largo de los siglos XVIII, XIX y principios del XX aparecen algunas obras que pueden relacionarse con el campo de la TA; como común denominador presentan la preocupación por crear un lenguaje universal, esta idea se refleja en obras como la de *Couturat y Leau* en su "Historia de la Lengua Universal" en 1903, en la que aparece el artículo de *W. Rieger*, titulado *Code Grammar which with help dictionaries enables the mechanical from one language into all others* o bien las bases de lo que actualmente es el esperanto²⁷.

En lo que al esperanto se refiere, en 1887, en Polonia, el doctor *Luís Lázaró Zamenhof* presentó su proyecto de lengua internacional, a la cual denominó *Esperanto*. Es un idioma planificado, con carácter internacional y neutro y es también un movimiento social. Está integrado por elementos tomados de entre los más difundidos en todas las lenguas. Estos elementos se estructuran con una gramática absolutamente lógica, libre de excepciones e irregularidades, por lo que el aprendizaje básico de esta lengua se lleva a cabo en tres o cuatro meses²⁸.

Estos empeños de ofrecer herramientas para la solución al problema de la diversidad de idiomas preceden por bastante tiempo a la propia existencia del computador. Por ello, se puede entender que desde el momento en que un computador estuvo disponible en la década de 1940, la TA pasó a convertirse inmediatamente en una de las aplicaciones estrella de la informática. Desde entonces, ha dado tiempo a realizar numerosos experimentos, pequeños y grandes, así como inversiones institucionales e industriales sustanciosas.

Se pueden identificar tres periodos en los que hubo apoyo para la investigación en el área. En Estados Unidos durante los años 50's y 60's, temerosos del poderío tecnológico soviético (particularmente después del lanzamiento del *Sputnik*²⁹ en 1.957), se estimuló el apoyo gubernamental y militar para la traducción del ruso al

²⁶ WILKINS, JOHN J. W. "An Essay Towards a Real Character and A Philosophical Language". 1 Jun 2005. <<http://www.thoemmes.com/language/wilkins.htm>>.

²⁷ "La traducción automática". 1 Jun 2005. <<http://www.dgbiblio.unam.mx/servicios/dgb/publicdgb/bole/fulltext/vollV3/traduccion.htm>>.

²⁸ "Esperanto: la lengua planetaria. Rumbo a la sociedad-mundo". 1 Jun 2005. <<http://www.monografias.com/trabajos18/esperanto/esperanto.shtml>>.

²⁹ El satélite Sputnik fue el primer satélite artificial lanzado por Rusia.

inglés. En los años 70's, los problemas causados por el multilingüismo de la Comunidad Europea, provocó el patrocinio de la investigación alrededor del sistema de traducción para el manejo de la documentación administrativa, técnica y económica entre las lenguas de los países miembros de la Comunidad Europea. En los años 80's en el proyecto japonés de la quinta generación, la TA jugó un papel muy importante, lo que llevó a este país a alcanzar una posición privilegiada en la investigación acerca de la TA.

Un referente obligado para conocer con más detalle la evolución de la TA es el académico británico *John Hutchins*³⁰, cuya bibliografía puede ser, consultada libremente en Internet.

En esta reseña de la TA se seguirá el esquema que aborda la historia de la TA por décadas.

2.7.1. Precursores y pioneros 1933 - 1954

En 1933 se solicitaron, una en Francia y otra en Rusia, dos patentes para *máquinas de traducir*. Por un lado, la del francoarmenio *Georges Artsouni*, que había diseñado un dispositivo de almacenaje en banda de papel con el que podía encontrarse el equivalente de cualquier palabra en otra lengua, de la cual, aparentemente un prototipo fue demostrado en 1937; y por otro lado, la del estudiante ruso *Petr Smirnov-Troyanskii*. De las dos, la considerada más significativa desde la perspectiva actual, es la segunda, puesto que los principios de su máquina siguen vigentes: las tres fases que establece en el proceso de traducción se pueden asimilar actualmente a las de análisis, transferencia y generación. Su sistema era muy rudimentario (pues sólo se planteaba la segunda fase como un proceso automático), pero lo cierto es que se adelantó a la época con sus ideas sobre TA³¹.

2.7.2. Primera década

Los primeros desarrollos informáticos reseñables se realizaron en el famoso computador *ENIAC*³² en 1946. Entre las investigaciones pioneras se encuentra la realizada por *Warren Weaver*³³ de la Fundación Rockefeller en 1947, cuyo resultado es el famoso documento conocido como *Wearever's Memorandum* publicado en

³⁰ HUTCHINS, John J. H. Op. Cit.

³¹ HERNANDEZ, Pilar P. H. "En torno a la traducción automática" El concepto de la traducción automática (2002).

³² ENIAC Electronic Numerical Integrator And Computer. En español: Computador e Integrador Numérico Electrónico.

³³ WEAVER, Warren W. W. Matemático estadounidense (Reedsburg, 1894 - New Milford, 1978)

1949³⁴, presentado en la primera conferencia sobre TA. Él fue quien dio a conocer públicamente la disciplina anticipando posibles métodos científicos para abordarla: el uso de técnicas criptográficas, la aplicación de los teoremas de *Shannon*³⁵ y la utilidad de la estadística, así como la posibilidad de aprovechar la lógica subyacente al lenguaje humano y sus aparentes propiedades universales. El mundo salía de una guerra mundial que en el plano científico había incentivado el desarrollo de métodos computacionales para descifrar mensajes en clave. A *Weaver*³⁶ se le atribuye haber dicho "cuando veo un artículo escrito en ruso me digo, esto en realidad está en inglés, aunque codificado con extraños símbolos. ¡Vamos a decodificarlo ahora mismo!". No hace falta decir que tanto los computadores como las técnicas de programación de aquellos años eran muy rudimentarias (se programaba mediante el cableado de tableros en lenguaje máquina), por lo que las posibilidades reales de probar los métodos eran mínimas.

2.7.3. Segunda década

Edwin Reifler de la *Universidad de Washington en Seattle*, en 1950, presenta avances en el campo de la TA al introducir conceptos de trabajo para la **pre-edición** y la **post-edición** en los resultados de una traducción, es aquí donde empiezan a considerarse las dificultades que resultan de la traducción palabra por palabra, olvidándose del contexto en que están inscritas³⁷.

En 1951 el prestigioso *Instituto de Tecnología de Massachusetts (MIT)*³⁸ puso a uno de sus especialistas - *Yehoshua Bar-Hillel*³⁹ - a trabajar con dedicación exclusiva en TA. Un año más tarde se organizó el primer simposio de la TA, con temas como los lenguajes controlados, los sublenguajes, la necesidad de la sintaxis, o la posibilidad de prescindir de la intervención humana. El crecimiento de los proyectos de investigación en la TA fue rápido entre 1956 y 1964.

El primer sistema de TA serio fue usado durante la guerra fría, para analizar textos rusos publicados en periódicos científicos. Las traducciones rudimentarias que se producían eran suficientes para entender el contexto de los artículos. Si un artículo discutido tenía un tema que demandaba ser analizado más profundamente era

³⁴ "Warren Weaver Memorandum" *MT News International* Jul 1999.

³⁵ SHANNON, Claude Elwood C. S. (Michigan, 1916 - 2001) recordado como "el padre de la teoría de la información".

³⁶ WEAVER, Warren W. W Op. Cit.

³⁷ "La traducción automática". 1 Jun 2005. Op. Cit.

³⁸ MIT (Massachusetts Institute of Technology) Instituto estadounidense dedicado a la ciencia, la ingeniería y la investigación.

³⁹ BAR-HILLEL, Yehoshua Y. B. (Viena, 1915 – Jerusalem 1975) Filósofo, matemático y lingüista.

enviado a un traductor humano para una traducción más completa, sino, era descartado.

La primera demostración pública de un traductor automático se llevó a cabo en 1954, en la *Universidad de Georgetown*, con ayuda de IBM⁴⁰. Se seleccionaron cuidadosamente 49 oraciones en ruso que se tradujeron al inglés con un vocabulario de 250 palabras y 6 reglas gramaticales. El éxito mediático de la demostración fue notable y en Estados Unidos se dedicaron importantes partidas presupuestarias (la mayoría aportadas por el Ministerio de Defensa) para traducir del ruso, francés y alemán al inglés. Fue un momento de euforia inicial, que llevó a plantear el objetivo de la *Fully Automatic High Quality Translation (FAHQT)*⁴¹. Entre los desarrollos pioneros de aquella década hay que destacar los de las universidades de *Georgetown* y *Texas*, donde se establecieron las bases de dos sistemas que todavía perduran, **SYSTRAN**⁴² y **METAL**⁴³ respectivamente.

En esta época, se mantuvieron activos muchos grupos. Unos dedicaron sus esfuerzos a crear sistemas que resultasen operativos en poco tiempo, otros se entregaron a la investigación teórica, realizando importantes aportaciones a la teoría lingüística. Los primeros sistemas creados, conocidos como **sistemas de primera generación**, consistieron principalmente en diccionarios bilingües muy extensos, donde las entradas daban lugar a una o más salidas, y alguna regla para producir el orden correcto de las palabras, si bien, con resultados muy limitados.

Así mismo, aparecen en el escenario varias universidades de diferentes países entre las que destacan: la *Universidad de Harvard*, la *Universidad de Michigan*, el *Instituto Lingüístico de Moscú, Leningrado y Praga*, la *Universidad de Tokio*, la *universidad de Texas*, la *Universidad de Montreal*, *Grenoble* y *Milán*, y al final de este periodo empieza a hacerse presente la influencia de corporaciones, tales como *IBM*, *Hitachi*, *Fujitsu*, *Colgate* y otras, en los proyectos de investigación y desarrollo del área.

⁴⁰ IBM (International Business Machines Corporation). Conocida coloquialmente como “el Gigante Azul”, es una empresa que fabrica y comercializa hardware, software y servicios relacionados con la informática. Tiene su sede en Armonk (Estados Unidos).

⁴¹ FAHQT En español: traducción de alta calidad totalmente automática.

⁴² SYSTRAN (SYStem TRANsaltion). Sistema de traducción creado por Peter Toma en 1970.

⁴³ METAL (MEchanical Translation and Analysis of Language).

2.7.4. Tercera década

Pero las considerables inversiones iniciales no daban los frutos deseados. Los investigadores antes entusiastas del método directo, convinieron, tras la presentación de *Chomsky*⁴⁴ en 1957 de un modelo formal de descripción lingüística, en que una traducción de cierta calidad sólo sería posible con un análisis sintáctico y semántico completo de las lenguas de origen y de destino. *Bar-Hillel*⁴⁵ en 1960 se atrevió a cuestionar la idea de la **FAHQT** aduciendo que para obtener resultados equiparables a los de la traducción humana habría que incorporar conocimiento semántico y pragmático en proporciones todavía no alcanzables, por lo que recomendó rebajar los objetivos. La euforia inicial fue dando paso, poco a poco, a un clima de desilusión, dada la complejidad de los problemas lingüísticos que se planteaban y la certeza de que los sistemas nunca llegarían a tener el conocimiento del mundo que tiene un ser humano. En 1964 el **National Research Council**⁴⁶ constituyó un comité, **ALPAC** (*Automatic Language Processing Advisory Committee*)⁴⁷, para evaluar la situación de la TA. Las conclusiones que se publicaron dos años más tarde tuvieron efectos demoledores: "... no se ha obtenido TA para textos científicos genéricos, y tampoco parece que se vaya a obtener a corto plazo". El resultado fue un drástico recorte financiero que literalmente terminó con la investigación en Estados Unidos durante más de diez años. Pese a ello, fue una época de enormes avances en el plano teórico. *Noam Chomsky*⁴⁸ revolucionó el estudio de las lenguas con la publicación de su *Syntactic Structures* en 1957. En el campo de la informática, nuevos diseños de estructuras de datos y lenguajes de programación de alto nivel como **ALGOL**⁴⁹ y **LISP**⁵⁰ llevaron al desarrollo de algoritmos y metodologías modulares que han sido fundamentales en la evolución de la disciplina.

Entre 1967 y 1977, la investigación sobre TA se desarrolló sobre todo fuera de Estados Unidos, concretamente en Canadá, Europa occidental y Japón, respondiendo a necesidades distintas con respecto al periodo anterior. La política bicultural desarrollada en Canadá originaba una elevada demanda de traducción inglés-francés que el mercado no podía satisfacer. Por su parte, en la **CE** ya

⁴⁴ CHOMSKY, Avram Noam N. C. (Pensilvania, 1928) Profesor de lingüística en el MIT (Instituto de Tecnología de Massachusetts).

⁴⁵ BAR-HILLEL, Yehoshua Y. B. Op. Cit.

⁴⁶ NRC (National Research Council) En español: Consejo de Investigación nacional. Es la principal organización del gobierno canadiense en cuanto a investigación y desarrollo, esta en funcionamiento desde 1916.

⁴⁷ ALPAC (Automatic Language Processing Advisory Committee) En español: Comité supervisor del procesamiento de lenguaje natural.

⁴⁸ CHOMSKY, Avram Noam N. C. Op. Cit.

⁴⁹ ALGOL (ALGOritmic Language) En español: lenguaje algorítmico.

⁵⁰ LISP (LISt Processing) En español: Procesamiento de listas.

entonces era necesario traducir gran cantidad de documentación legal, administrativa, técnica y científica, que comprendía todas las lenguas de la comunidad.

2.7.5. Cuarta década

El informe **ALPAC** afectó dramáticamente a la TA en Estados Unidos, pero en Canadá y Europa apenas tuvo incidencia. En 1976 investigadores del grupo **TAUM** (*Traduction Automatique de l'Université de Montréal*)⁵¹ presentaron el sistema **MÉTÉO**⁵², que traducía informes meteorológicos del inglés al francés. Es un sistema que ha hecho historia, por la idoneidad de la aplicación y diseño. Ese mismo año la CE decidió recurrir a la TA para hacer frente a la desbordante demanda de traducciones internas en sus diversas sedes administrativas. La Comisión compró las licencias para desarrollar SYSTRAN y adaptarlo a sus necesidades. Poco después, con la idea de impulsar la investigación en Europa y elevar la calidad de las traducciones, la propia Comisión financia el ambicioso proyecto **EUROTRA**⁵³, el cual, hasta 1992 ocuparía a numerosos investigadores de todos los países miembros, el fruto de dicho proyecto debería ser un programa para todas las lenguas comunitarias. Esta etapa, que abarca todos los ochenta, se caracterizó por un fuerte desarrollo de los métodos simbólicos y una gran vitalidad de la investigación en sintaxis (gramáticas basadas en la unificación de rasgos) y en semántica (formalismos basados en la lógica de predicados). Sin embargo, los avances en el plano teórico no acababan de trasladarse al terreno de los resultados.

Así mismo, en los años 80 emergieron nuevos sistemas, a la vez que crecía el número de países implicados en la investigación. Cabe destacar los sistemas SYSTRAN (el cual ahora opera con muchos pares de lenguas), METAL, los sistemas internos de TA desarrollados por la *Organización Panamericana de Salud*⁵⁴ y también los sistemas de traducción del japonés al inglés y viceversa de compañías japonesas de computadores. La amplia disponibilidad de microcomputadores y de software de procesadores de texto creó un mercado para sistemas de TA más económico. Estos sistemas fueron explotados por compañías

⁵¹ TAUM (*Traduction Automatique de l'Université de Montréal*) En español: Traducción Automática de la Universidad de Montreal.

⁵² MÉTÉO vocablo francés que significa “clima”

⁵³ EUROTRA tuvo como objetivo la creación de un prototipo de laboratorio para la traducción automática a los nueve idiomas de la Unión Europea.

⁵⁴ PAHO (*The Pan American Health Organization*)

europeas y norteamericanas como **ALPS**⁵⁵ y **GLOBALINK**⁵⁶, entre otras, y japonesas como **NEC**⁵⁷ y **Sanyo**⁵⁸. En 1980 **Philips**⁵⁹ da inicio a su proyecto **Rosetta**⁶⁰.

En los años comprendidos entre 1984 y 1988 **Ntran**⁶¹ desarrolló **UMIST**⁶² en el centro de lingüística computacional. Su primer prototipo fue desarrollado en PROLOG⁶³.

2.7.6. Quinta década

Producto de las conclusiones del *informe Danzin*⁶⁴, encargado en 1991 por la CE, hubo en Europa dos malas noticias relacionadas con la TA. Por el lado institucional, la CE decidió cancelar definitivamente la financiación de EUROTRA; por el lado empresarial, PHILIPS inesperadamente da por terminado uno de los proyectos de más prestigio entre los especialistas, Rosetta. Paralelamente en Japón, se aplica una política de moderación presupuestaria tras las fabulosas inversiones de los años precedentes. En este contexto de declive generalizado, hace su aparición en el mercado un nuevo tipo de producto de traducción asistida, de diseño muy distinto a los anteriores. Son los programas de gestión de **memorias de traducción**, dados a conocer primeramente por IBM que desarrolló *TranslationManager*⁶⁵ y posteriormente llevados al gran público por las empresas alemanas **TRADOS**⁶⁶ con su desarrollo *Translator's Workbench*⁶⁷ y **STAR**⁶⁸ con

⁵⁵ ALPS (Active Learning Practice for Schools) En español: Práctica Activa del Aprendizaje para Instituciones Educativas, es una comunidad electrónica dedicada a la mejora y al adelanto de la enseñanza y de las prácticas educativas.

⁵⁶ GLOBALINK es un programa de traducción automática de textos basado en algoritmos lógicos de conversión lingüística y análisis morfológico de frases.

⁵⁷ NEC (Nippon Electric Company) En español: Compañía Eléctrica Japonesa. En 1983, fue retitulada NEC Corporation.

⁵⁸ Sanyo es una compañía japonesa que fabrica una amplia gama de electrodomésticos.

⁵⁹ Philips, empresa de productos eléctricos y electrónicos creada en Holanda por Royal Philips.

⁶⁰ Rosetta, es un proyecto de colaboración global entre especialistas de la lengua y los nativos que construyen un archivo en línea público accesible para todos los idiomas humanos conocidos.

⁶¹ Ntran, proyecto de TA de inglés-japonés para usuarios cuya lengua nativa fuera el inglés.

⁶² UMIST (University Of Manchester Institute Of Science And Technology) En español: Instituto de Ciencia y Tecnología de la Universidad de Manchester

⁶³ PROLOG (PROgramming in LOGic) Lenguaje de programación declarativo utilizado en Inteligencia artificial.

⁶⁴ Informe Danzin, Informe evaluador de la traducción automática encargado por la CE.

⁶⁵ TranslationManager, Memoria de traducción desarrollada por IBM.

⁶⁶ TRADOS, Empresas alemanas desarrolladoras de software de traducción automatizada, herramientas de traducción y memoria de la traducción.

⁶⁷ Translator's Workbench, Memoria de traducción basada en una sofisticada base de datos.

⁶⁸ STAR Servicios Lingüísticos se fundó en 1999 en Barcelona como parte del grupo STAR.

TRANSIT⁶⁹, y la española **ATRIL**⁷⁰ con el conocido producto **DÉJÀ-VU**⁷¹. Otro aspecto destacable de esta etapa es el desarrollo de Internet, así como el cambio de enfoque de la traducción hacia la localización.

Por último, desde los años 90 hasta hoy se han dado importantes pasos hacia adelante en la investigación sobre traducción del discurso, y los esfuerzos de numerosos grupos de investigación se han orientado a la creación de aplicaciones prácticas. De ello, el ejemplo más significativo es tal vez, la creación de estaciones de trabajo para traductores profesionales. La TA, como instrumento de masa, comienza a dar sus primeros pasos con la aparición de numerosos programas (encaminados a satisfacer las necesidades de un cada vez más amplio abanico de potenciales usuarios) y el desarrollo de proyectos ni siquiera soñados hace unas décadas. Esta toma de conciencia de la importancia de las herramientas de traducción no es sino un claro índice del saludable estado en que se encuentra la investigación, aunque esto no excluye que puedan seguir existiendo ideas equivocadas al respecto. Es importante hacer una referencia hacia el proyecto de TA del investigador japonés *Hiroshi Uchida*⁷², cuya presentación oficial tuvo lugar en Barcelona en 2004: el Lenguaje Universal para la Red o UNL⁷³.

2.7.7. Sexta década

En la actualidad la humanidad esta ante el comienzo de una etapa nueva. La globalización de empresas y mercados lleva a la par la necesidad de adaptar localmente productos y servicios. Herederas del pasado, un buen número de tecnologías se están integrando y complementando en arquitecturas híbridas, que conciben la traducción como un eslabón más en el complejo ciclo de la información en un medio que ahora sí es electrónico. El lugar hacia donde convergen todos estos desarrollos es Internet, con implicaciones que todavía no se pueden anticipar, pero que en gran medida dependen de cómo se resuelvan la propiedad intelectual y los derechos de explotación de los recursos lingüísticos que se van acumulando.

El nuevo sistema de evaluación **BLEU**⁷⁴, propuesto por IBM en 2002, ha significado un nuevo impulso a la dinamización de la investigación. El mejor

⁶⁹ TRANSIT, constituye la herramienta principal de los productos STAR.

⁷⁰ ATRIL, empresa desarrolladora de software de traducción.

⁷¹ DÉJÀ-VU, Memoria de Traducción desarrollada por ATRIL.

⁷² UCHIDA, HIROSHI H. U. Director de UNDL (Universal Networking Digital Language) Foundation.

⁷³ "Universal Networking Language". 2005. 1 Jun 2005. <<http://www.undl.org/unlsys/unl/unl2005/main.htm>>.

⁷⁴ PAPINENI, KISHORE K. P., ROUKOS, SALIM S. R., WARD, TODD T. W., and ZHU, WEI-JING W. J. Z. 2001. Bleu: a Method for Automatic Evaluation of Machine.

exponente es la competición auspiciada por el instituto **NILS**⁷⁵ del gobierno de Estados Unidos. Los sistemas que compiten están basados en la estadística, que se constituye en el enfoque que actualmente mejores expectativas genera.

2.7.8. Historia y antecedentes de SYSTRAN

SYSTRAN se trata de un sistema de TA⁷⁶. *Peter Toma* su creador, desarrolló un primer sistema en la *Universidad de Georgetown*, que con el paso de los años fue perfeccionando hasta dar con el definitivo **SYSTRAN**. Él fue además el encargado de traducirlo del ruso al inglés para las fuerzas aéreas americanas. Más tarde, la *NASA*⁷⁷ probó el sistema en uno de sus proyectos, y aunque no dio muy buenos resultados, esta experiencia otorgó renombre a **SYSTRAN**. Tras la fama recién adquirida, la *CE* solicitó a *Peter Toma* una demostración entre el par inglés/francés, tras la cuál quedó satisfecha. En el año 1976, **SYSTRAN** fue comprado por la *CE* y posteriormente desarrollado. Fue el burócrata *Loll Rolling*⁷⁸ el que introdujo la TA en la *CE* ya que obtuvo la licencia para usar **SYSTRAN**. Éste es el motivo de que los diccionarios del sistema se hayan llenado con terminología propia de la *CE*. En la actualidad el sistema de la *CE* dispone de 17 pares de lenguas que se han integrado a una red local de servicios lingüísticos. Actualmente las traducciones se procesan a una velocidad de 500.000 palabras por hora.

A principios de los noventa, una empresa francesa llamada **GACHOT**⁷⁹ adquirió todas las filiales, salvo la de la *CE*, y el sistema se hizo muy popular en Francia debido a su accesibilidad. En 1994 **SYSTRAN** se ofrece de manera gratuita en los Chat de *CompuServe*⁸⁰, y un año más tarde se creó una versión adaptada para *MS-Windows*⁸¹. Pero el éxito definitivo de **SYSTRAN** se produjo en 1997, cuando tras firmar un acuerdo con **AltaVista**⁸² se ofrecía el servicio de traducción **BABEL FISH**⁸³ de manera gratuita. Hoy en día **SYSTRAN** es el sistema de traducción más

⁷⁵ NILS (National Information and Library Services) En español: Servicios Nacionales de Información y Biblioteca.

⁷⁶ SYSTRAN (SYStem TRANslation). Op. Cit.

⁷⁷ NASA. (National Aeronautics and Space Administration). Agencia gubernamental de los Estados Unidos responsable de los programas espaciales.

⁷⁸ ROLLING, LOLL L. R. Jefe de división de ingeniería lingüística en la *CE*.

⁷⁹ GACHOT. Empresa que ofrece los servicios de Systran a través del Minitel francés, ostenta los derechos mundiales en el sector privado.

⁸⁰ CompuServe. Empresa estadounidense perteneciente a AOL (American OnLine) iniciadora de la comercialización generalizada del acceso a Internet vía telefónica.

⁸¹ MS-Windows. Sistema operativo creado por Microsoft.

⁸² AltaVista. Proveedor de tecnología y servicios de búsqueda creado en 1995 por los científicos del Laboratorio de investigaciones de Digital Equipment Corporation en Palo Alto, California.

⁸³ Babelfish. Servicio de traducción mecanizada de la Web propiedad de AltaVista.

desarrollado con 35 pares de lenguas disponibles y el más utilizado con 1.000.000 de traducciones a través de BABELFISH.

2.8. LA TA A NIVEL MUNDIAL

2.8.1. Sistemas de TA utilizados a nivel mundial

Los sistemas que cubren los idiomas principales del occidente europeo (inglés, francés, alemán, italiano, portugués y español) incluyen a *Systran Enterprise*, *LogoMedia Enterprise*, *SDL Enterprise Translator* e *IBM WebSphere*. Todos estos son sistemas cliente/servidor diseñados especialmente para servicios de traducción de grandes empresas. Otros tipos de sistemas cubren el inglés e idiomas del lejano oriente (japonés, coreano y chino). Aquí se encuentran a *Fujitsu ATLAS*, *EWTranslate*, *IBM WebSphere* y *Systran Enterprise*. Otros sistemas cubren árabe e inglés como *TranSphere* de *AppTek* y finlandés e inglés como *TranSmart* de *Kielikone*. El dominio del inglés sale a relucir al aparecer en todos los sistemas como idioma fuente y/o destino. Otras combinaciones son menos comunes, como francés y alemán, o simplemente no se han tenido en cuenta, como en el caso de holandés e italiano, español y japonés. Muchos idiomas no son representados por completo en sistemas empresariales, simplemente por que no tienen suficiente demanda comercial.

2.8.2. Análisis del compendio de la EAMT de software de traducción

La Asociación Europea para la Traducción Automática **EAMT**⁸⁴, dos o tres veces por año realiza una publicación de un compendio en el que realiza un análisis de los diferentes productos en el mercado de herramientas de software dedicadas a la TA, la octava edición (Accesible públicamente) de dicho compendio compilado por *John Hutchins* en compañía de *Walter Hartmann* y *Etsuo Ito*, fue publicado en enero de 2004⁸⁵. Es importante resaltar que en el compendio no se establece preferencia por ninguna de las herramientas ni fabricantes. Sólo se limitan a ofrecer información de la cual se pueden extraer los siguientes datos.

2.8.2.1. Pares de idiomas

Existen 160 diferentes combinaciones de idiomas para las que existen herramientas de traducción, se destacan el inglés con 37 pares de idiomas, el

⁸⁴ EAMT (European Association for Machine Translation). Es una organización sin ánimo de lucro registrada en Suiza.

⁸⁵ HUTCHINS, John J. H., HARTMANN, Walter W. H., and ITO, Etsuo E. I. EUROPEAN ASSOCIATION FOR MACHINE TRANSLATION. Op. Cit.

alemán con 22 y el japonés con 12. El español posee nueve combinaciones (inglés, francés, alemán, japonés, italiano, catalán, ruso, coreano y portugués) (Véase el Anexo D).

2.8.2.2. Combinaciones con diccionario electrónico

Existen 146 combinaciones para las cuales se cuenta con un diccionario electrónico. Es importante destacar que en realidad existen muchas más combinaciones, ya que para idiomas en los cuales existen diccionarios multilingües el compendio en la mayoría de ocasiones no especifica cuales o cuantos son los idiomas que maneja (Véase el Anexo D). De estas 17 combinaciones involucran al idioma español (Véase el Anexo D).

2.8.2.3. Número de Productos en el mercado

De las tres categorías generales y las 17 subcategorías nombradas anteriormente, el compendio contiene un total de 408 productos, resaltando además que en este compendio no se tienen en cuenta aquellos productos que están en desarrollo o que han reportado cierre hacia el mercado, ni tampoco aquellas herramientas de disponibilidad limitada, tal como sistemas desarrollados para clientes particulares.

2.8.2.4. Cantidad de empresas en el mercado

La lista de Compañías listadas en el compendio se incluye productores, vendedores, distribuidores y proveedores de servicios de sistemas de TA. Se listan un total de 198 empresas conocidas (Véase el Anexo D).

2.9. DESCRIPCIÓN FUNCIONAL DE PRODUCTOS EXISTENTES

Entre los productos existentes en el mercado que están relacionados con la traducción, se puede encontrar software de TA, software de soporte para la traducción y servicios de traducción.

2.9.1. Software de TA

Por software de TA se entiende las aplicaciones de TA que realiza el proceso de traducción sin intervención humana.

2.9.1.1. Sistemas TA (Generales)

Software de TA, donde se tiene una entrada de un texto en un idioma y se produce la salida de este en otro idioma.

2.9.1.2. Sistemas TA (Uso doméstico))

Un sistema TA para uso público y general, algunos de estos son solamente algo más que pequeños diccionarios.

2.9.1.3. Sistemas TA (Internet y Web)

Sistemas desarrollados específicamente para la traducción de documentos electrónicos en Internet.

2.9.1.4. Sistemas TA (Uso profesional)

Sistemas diseñados para traductores con experiencia.

2.9.1.5. Sistemas TA (Cliente/Servidor)

Sistemas desarrollados para una intranet.

2.9.2. Herramientas para el soporte de traducción

Son herramientas de software que ayudan al traductor humano en el proceso de traducción, permitiéndole automatizar algunas de las tareas o brindándole información requerida en el proceso.

2.9.2.1. Diccionario electrónico

El diccionario electrónico se refiere generalmente a una base de datos bilingüe o multilingüe de entradas léxicas (palabras o frases), el cual generalmente se encuentra digitalmente. Soportando casi siempre la característica de hipertextualidad. Muchos de estos diccionarios electrónicos no son más que adaptaciones al formato digital de versiones anteriores en formato en papel (es el caso, entre muchos otros, de las versiones electrónicas del *DRAE*⁸⁶ y del *DUE*⁸⁷, etc.). Los diccionarios en *CD-ROM*⁸⁸ o los diccionarios en línea (ciberdiccionarios) han mejorado la sistematización, coherencia, presentación de las distintas informaciones lingüísticas, la rapidez y variedad de consulta, rasgos condicionados por la estructura de la base de datos. Además, el manejo de los cuerpos de texto y de las concordancias permite refinar la selección de los lemas,

⁸⁶ DREA (Diccionario de la Real Academia Española)

⁸⁷ DUE (Diccionario del Uso de Español)

⁸⁸ CD-ROM (Compact Disc-Read Only Memory) En español: Disco compacto de memoria de solo lectura.

las definiciones, la organización de las acepciones y la ejemplificación de los artículos.

Actualmente los diccionarios electrónicos han crecido ampliamente, permitiendo encontrar inclusive diccionarios especializados en temas como informática, telecomunicaciones, biología y en muchos casos diccionarios y glosarios de lenguajes de programación como *Java*⁸⁹ y *C++*⁹⁰, entre otros, e inclusive diccionarios dedicados a definiciones de fútbol y otros deportes.

Algo esencial en un diccionario para la TA es la cantidad de información que puede brindar ya que como lo describe *Doug Arnold* en el libro "Machine Translation: An Introductory Guide"⁹¹, no basta con que brinde únicamente la traducción de una palabra de un idioma a otro sino que además de esto deberá incluir información gramatical y semántica que posteriormente permitirá realizar los análisis sintáctico, morfológico y semántico, los cuales son indispensables para obtener una buena traducción.

2.9.2.2. Herramientas para soporte de localización

La localización consiste en traducir medios electrónicos como sitios *Web* y *software*. La localización también puede ser extendida a traducir la documentación, materiales de publicidad, etc. Esencialmente, la localización implica el adaptar el material para cumplir los requerimientos lingüísticos, culturales y técnicos del nicho de mercado destino.

Las herramientas de localización son aplicaciones que ayudan a los especialistas de localización en la traducción de medios electrónicos.

2.9.2.3. Sistemas de memorias de traducción

Las memorias de traducción son programas informáticos diseñados para ayudar a los traductores. Las memorias de traducción suelen utilizarse en combinación con un procesador de texto, un sistema de gestión de terminología, un diccionario multilingüe, e incluso los resultados de una TA sin pulir.

⁸⁹ Java. Lenguaje de programación creado por Sun Microsystems.

⁹⁰ C++. Lenguaje de programación, diseñado a mediados de los ochenta, por Bjarne Stroustrup, como extensión del lenguaje de programación C.

⁹¹ ARNOLD, Doug D. A. "Machine Translation: an Introductory Guide" Op. Cit.

Una memoria de traducción consta de una base de datos de segmentos de texto en una lengua fuente y sus traducciones en una o más lenguas destino. Estos segmentos pueden ser palabras sueltas o frases.

Al utilizarlas, el traductor abre un texto fuente (el texto que se va a traducir) y el programa intenta encontrar segmentos en la lengua destino para generar un texto parcialmente traducido.

Algunas memorias de traducción sólo buscan correspondencias literales, es decir, sólo pueden recuperar coincidencias exactas de las frases, mientras que otras utilizan algoritmos para encontrar cadenas de texto similares (*fuzzy matches*) en las que marcan las diferencias. La flexibilidad y la solidez del algoritmo determinan en gran medida el rendimiento de la memoria de traducción, aunque para algunos usos la tasa de recuerdo de coincidencias exactas puede ser lo suficientemente alta como para justificar el enfoque literal.

El traductor debe traducir de forma manual los segmentos para los que no se encuentre coincidencia, segmentos que se almacenan en la base de datos para utilizarse en futuras traducciones.

Las memorias de traducción funcionan de forma óptima con textos muy repetitivos, como los manuales técnicos. También son útiles para realizar adiciones a textos correspondientes, por ejemplo, a pequeñas modificaciones de productos. Utilizar una memoria de traducción de forma reiterada en los textos adecuados durante algún tiempo puede ahorrar mucho trabajo a un traductor. Las memorias de traducción no son adecuadas para textos literarios o creativos por el simple hecho de que hay muy poca repetición en el lenguaje empleado.

El concepto de memoria de traducción no es nuevo (tiene más de veinte años), pero sólo recientemente ha adquirido una entidad comercial significativa.

2.9.2.4. Herramientas de alineamiento

Software para creación de bases de datos bilingües donde las frases corresponden a segmentos de texto del lenguaje destino.

2.9.2.5. Sistema de manejo de terminología

Software para creación, mantenimiento y búsqueda de bases de datos multilingües de terminología compilada a nivel local para uso personal o de compañías. En otras palabras, se trata de un diccionario especializado con un mayor grado de complejidad.

2.9.2.6. Herramientas de preedición

Software para la preparación de textos de entrada, significa algo así como un control en el texto de entrada.

2.9.2.7. Estaciones de traducción

Sistemas integrados para traductores profesionales, los cuales combinan normalmente, procesamiento de palabras multilingüe, manejo de terminología, memorias de traducción y TA (opcional).

2.9.3. Servicios de traducción

Los servicios de traducción normalmente son sistemas de TA que se encuentran alojados en un sitio o portal *Web*.

2.9.3.1. Portal de TA

Servicios en Internet que proveen acceso a un número de sistemas o servicios de TA o información acerca de estos.

2.9.3.2. Servicio de TA

Servicio de traducción vía Internet, usando sistemas de TA. Muchos de ellos son gratuitos; algunos cobran de acuerdo a la longitud y tema de los textos y la post edición que se requiera.

Un servicio de TA se trata generalmente de un sitio *Web* que soporta TA, en la mayoría de los casos la interfaz *Web* se compone principalmente de tres cosas:

Una casilla de selección de idiomas, en donde se puede seleccionar los pares de idiomas (fuente y destino) de la traducción. Un campo para la inserción del texto en el idioma fuente que será traducido al idioma destino. Un botón de confirmación, el cual deberá ser presionado una vez se haya insertado el texto para proceder con el proceso de traducción.

En el servidor en donde se encuentra el servicio de traducción generalmente está instalado un software de TA, el cual se encargará de dar soporte a este proceso.

En la mayoría de los casos tanto la entrada como salida de texto que usa el sistema deben estar en formato plano y en algunos casos dichos servicios pueden llegar a traducir páginas *Web*, formato PDF⁹² y otros formatos de entrada.

2.10. MERCADO DE LA TA

A pesar de sus limitaciones inherentes, los sistemas TA están siendo usados actualmente por varias organizaciones a través del mundo. Probablemente el usuario más grande es la CE, la cual usa una versión hecha a la medida del sistema TA comercial **SYSTRAN** para manejar la TA de un gran volumen de borradores de documentos de uso interno.

Así mismo fue revelado que en Abril de 2003 *Microsoft* empezó a usar un sistema TA híbrido para la traducción de una base de datos de documentos de soporte técnico del inglés al español por un grupo de desarrolladores de la empresa⁹³. El grupo actualmente se encuentra realizando pruebas de inglés a japonés, así mismo probando con sistemas en línea de inglés a francés e inglés a alemán. Los dos últimos sistemas usan un componente de generación de lenguaje aprendido mientras que los dos primeros poseen componentes de generación desarrollados manualmente. Los sistemas fueron desarrollados y probados usando una base de datos de mas de un millón de oraciones cada uno.

De acuerdo con un estudio de *ABI Research*⁹⁴, el sector de la traducción humana creará por sí solo ingresos por 11,5 billones de dólares hasta 2007, mientras que los ingresos relacionados con la TA llegarán a 134 millones de dólares durante el mismo periodo.

Además de eso, entre 2001 y 2007, el valor del mercado de la localización de software deberá aumentar de 1,1 a 3,4 billones de dólares. El valor de la localización de sitios en la *Web* conocerá, según se prevé, un crecimiento aún más espectacular, subiendo de 499 millones para 3,1 billones de dólares en seis años.

Según la *International Data Corporation*⁹⁵, el mercado mundial de equipos y software de administración de contenido deberá exhibir un crecimiento de 60 por ciento entre 2003 y 2007 para alcanzar cinco billones de dólares.

⁹² PDF (Portable Document Format). En español: Formato de Documento Portable.

⁹³ Microsoft Natural Language Research group. En español: Grupo Microsoft de Investigación del Lenguaje Natural.

⁹⁴ ABI-Research. Abastecedor de informes de investigación tecnológica.

⁹⁵ International Data Corporation. En español: Corporación Internacional de Datos.

Otras dos organizaciones estudiaron el sector del procesamiento de voz y llegaron a la conclusión de que sus ingresos subieron dramáticamente, entre 2002 y 2003, de cerca de 677 millones de dólares americanos según *ABI Research* y para 800 millones de dólares según *Datamonitor*⁹⁶. Las dos organizaciones prevén, además de esto, que estos ingresos pasarán a la suma imponente de cinco billones de dólares hasta 2008.

Hay diversos factores que explican este crecimiento:

- Expansión del comercio internacional.
- Globalización de las empresas.
- Evolución fulgurante de la Internet.
- La rápida evolución de las tecnologías de la información y de las telecomunicaciones.
- El número creciente de discusiones estratégicas en muchos lenguajes diferentes en relación a la seguridad en el mundo.

La industria del lenguaje está involucrada en todos los sectores de la economía, desde el turismo a las comunicaciones y a la tecnología de la información, pasando por el mercado de los bienes y servicios. Dada la limitante de espacio y porque además el objetivo del presente trabajo no es profundizar en cada uno de los temas afectados por la traducción automática, no se ahonda más en el tema. Aún así, en los anexos se ha incluido una introducción a una amplia variedad de este tipo de temas.

En el Anexo A se incluye información acerca de algunas tendencias de idiomas y culturas.

Con respecto a los problemas de la comunicación en las grandes empresas a nivel mundial se ha incluido el Anexo B.

Igualmente, en el Anexo C se tratan los aspectos culturales descritos por la Recomendación Sobre la Promoción y el Uso del Plurilingüismo y el Acceso Universal al Ciberespacio” publicado por la UNESCO⁹⁷.

⁹⁶ Datamonitor. Compañía líder en la provisión de información empresarial.

⁹⁷ United Nations Educational, Scientific and Cultural Organization. En español: Organización de las Naciones Unidas para la Educación la Ciencia y la Cultura.

Además, si se desea conocer algunos datos acerca de la demanda mundial de la traducción automática, se recomienda ver el Anexo D.

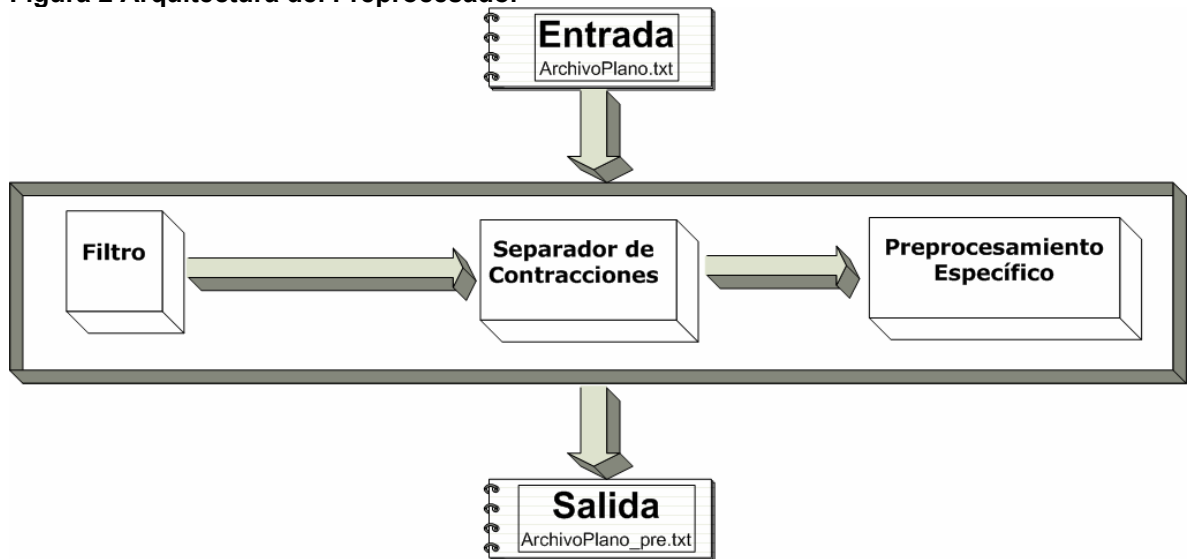
En el presente capítulo se ha tratado el estado del arte de la TA. Se ha realizado una introducción a esta área del conocimiento y se ha descrito de una manera general los desarrollos realizados. De igual manera, se ha descrito que dicho proceso consta de varias etapas. En los capítulos subsiguientes se profundizará en cada una de estas, realizando una introducción teórica, un modelamiento sobre la manera de atacar el problema y la descripción de la implementación de un prototipo que demuestra su funcionalidad.

3. PREPROCESAMIENTO

El objetivo del **preprocesador** es darle un formato al texto antes de ser sometido a los diferentes módulos de traducción. El **formateo** consiste en la eliminación de caracteres inútiles y adecuación del texto para hacer más fácil su tratamiento en las posteriores fases.

3.1. ARQUITECTURA DEL PREPROCESADOR

Figura 2 Arquitectura del Preprocesador



Como se puede ver en la Figura 2, el preprocesador está compuesto por tres módulos, un filtro, un separador de contracciones y un módulo dependiente del idioma que se encarga de realizar labores adicionales de preprocesamiento para los lenguajes fuente que así lo requieran.

3.1.1. Filtro

Mediante el empleo de expresiones regulares, el módulo de filtro es el encargado de compactar los separadores redundantes que existen en el texto, es decir, eliminar múltiples espacios, espacios a inicio de frase, etc⁹⁸. Igualmente, se hacen

⁹⁸ VILARES FERRO, Jesús. "Aplicaciones del procesamiento del lenguaje natural en la recuperación de información en español." Doctoral Thesis. UNIVERSIDAD DE LA CORUÑA, 2005.

tratamientos al texto en cuanto a la disposición de las palabras, lo que permitirá obtener un texto más normalizado, por ejemplo, unir en una sola palabra dos fragmentos separados por guión en un salto de línea. Hay que tener claro que en esta etapa solo se trata el formato del texto, es decir, disposición de las palabras y redundancia de separadores.

Podrían también incluirse, asimismo, procesos de conversión de otros formatos, por ejemplo, *HTML*, *XML*, *PDF*, *DOC*, etc. a texto plano, si bien dicha conversión podría también llevarse a cabo en una fase previa al preprocesamiento.

En la Figura 3 se muestra un ejemplo de un texto antes y después de ser preprocesado por el módulo de filtrado.

Figura 3 Texto antes y después de filtrar

Antes del Preprocesamiento	Después del Preprocesamiento
Brad Smith, Microsoft's general counsel , called the decision to license parts of the source code for Windows "a bold stroke" that should put to rest charges that the company is holding back clear and timely information from competitors. Mr. Smith also signaled a shift in the Microsoft business model, calling the decision "quite a substantial step, quite a significant change, from the steps we have made in the past."	Brad Smith, Microsoft's general counsel, called the decision to license parts of the source code for Windows "a bold stroke" that should put to rest charges that the company is holding back clear and timely information from competitors. Mr. Smith also signaled a shift in the Microsoft business model, calling the decision "quite a substantial step, quite a significant change, from the steps we have made in the past."

3.1.1.1. Modelamiento del módulo filtro

Como ya se indicó anteriormente para realizar el módulo de filtro de preprocesamiento se hace uso de expresiones regulares. De esta manera se definen los siguientes patrones de expresiones regulares:

- *Múltiples Enters:* $(\backslash n \backslash s^*)$. El cual busca espacios redundantes entre saltos de línea. Se entiende espacios como espacios en blanco saltos de línea y tabulaciones redundantes. Se hace necesario que por lo menos haya un salto de línea seguido de cero o más espacios $(\backslash s)$. Es importante resaltar que $\backslash s$ es igual a $[\backslash s \backslash n \backslash t]$.
- *Múltiples Espacios:* $[\backslash t \]^+$. El cual busca espacios diferentes a Enter esto para eliminar espacios redundantes entre palabras. Dichos espacios pueden ser tabulaciones o espacios en blanco.

- *Espacios Iniciales*: $(^[\t J+)$. El cual busca uno o más espacios (tabulaciones o espacios en blanco) seguidos en el comienzo de una línea.
- *Separación Guión*: $(-\backslash n)$. El cual busca un guión seguido de un salto de línea. Con este patrón se podrá realizar el proceso de unir en una sola palabra dos fragmentos separados por guión en un salto de línea.

Finalmente para realizar el proceso de filtrado se realiza una substitución de las cadenas que cumplan con los patrones anteriores de la siguiente manera:

- *Múltiples Enters*: $(\backslash n\backslash s^*) \rightarrow \backslash n$. Se sustituyen por un solo salto de línea, eliminando de esta manera saltos de línea y espacios redundantes entre saltos de línea.
- *Múltiples Espacios*: $[\t J+ \rightarrow ' '$. Se sustituyen por un solo espacio, eliminando de esta manera espacios redundantes.
- *Espacios Iniciales*: $(^[\t J+) \rightarrow ''$. Se sustituyen por una cadena vacía, eliminando así espacios al inicio de una línea.
- *Separación Guión*: $(-\backslash n) \rightarrow ''$. Se sustituyen por una cadena vacía, uniendo de esta manera en una sola palabra dos fragmentos separados por guión en un salto de línea.

3.1.1.2. Implementación del prototipo del módulo filtro

El módulo desarrollado para el filtro modifica el archivo de entrada el cual es un archivo de tipo texto (extensión *.txt*) y genera un archivo con extensión *.pff* (*Preprocessing Filtered File*)⁹⁹. El módulo es independiente del idioma, pues solo trabaja a nivel estructural (forma) sin procesar el contenido de la información.

A continuación se muestra el código del módulo de filtrado desarrollado en *Python*¹⁰⁰. Se importa *re* para el manejo de expresiones regulares y *Archivo* de *traductor.librerias.IO* para las operaciones de lectura y escritura de archivos. Lo que hace básicamente es leer el archivo de entrada, busca y une los fragmentos de palabras separados por un salto de línea, parte en palabras todo el

⁹⁹ En el presente trabajo, se ha creado una extensión para nombrar el archivo de salida en esta etapa. Esta extensión no es un estándar y el archivo generado aquí es usado temporalmente para crear una interfaz con el módulo de expansión de contracciones.

¹⁰⁰ Python. Es un lenguaje de programación interpretado e interactivo multiplataforma. Fue creado por Guido Van Rossum en 1990. <www.python.org>

archivo uniéndolas nuevamente logrando así eliminar espacios redundantes y por último escribe en el archivo de salida.

```
from traductor.librerias.IO import Archivo
import re

def _filtrar(self):
    salida = Archivo(self.archivo.get_nombreSolo() + '.pff')
    textoSalida = self.archivo.leer()
    textoSalida = textoSalida.replace('\n', '#n#')
    cadenas = textoSalida.split('-#n#')
    patCadInicial = '^[^ ]+'
    textoSalida = ""
    numCadenas = len(cadenas)
    cadenaActual = 0
    while cadenaActual + 1 < numCadenas:
        cadenaAnterior = cadenas[cadenaActual]
        cadenaPosterior = cadenas[cadenaActual + 1]
        cadPosteriorIni = ' '.join(re.findall(patCadInicial,
                                                cadenaPosterior))
        cadenas[cadenaActual + 1] = ' '.join(re.split(patCadInicial,
                                                         cadenaPosterior))

        textoSalida = textoSalida + ' ' + cadenaAnterior +
                        cadPosteriorIni + '#n#'
        cadenaActual = cadenaActual + 1
    textoSalida = textoSalida + ' ' + cadenas[-1]
    textoSalida = ' '.join(textoSalida.split())
    textoSalida = re.sub('#n#\s*', '\n', textoSalida)
    salida.escribir(textoSalida)
```

3.1.2. Contracciones

En la gramática tradicional, una contracción es la formación de una nueva palabra a partir de dos o más palabras individuales. Esto es a menudo el resultado de una secuencia común de palabras, o como en el francés para mantener un sonido fluido. Sin embargo, la contracción ha ganado un significado más amplio tanto en la lingüística como en otras áreas de la investigación del lenguaje. Basados en las últimas definiciones. La contracción es la abreviación de una palabra, sílaba o grupo de palabras por la omisión de letras internas¹⁰¹.

¹⁰¹ ISO 4:1984, Documentación – *Reglas para la abreviación de las palabras de los títulos y los títulos de las publicaciones.*

3.1.2.1. Modelamiento del módulo de contracciones

Tomando en cuenta la anterior afirmación se puede definir una contracción en términos generales de la siguiente manera:

W_n es una contracción de $W_1 W_2 \dots W_k$ si $W_1 W_2 \dots W_k$ se puede reescribir como W_k abreviadamente. Donde W_i es una palabra. Por ejemplo:

- *del* es una contracción de *de el* ya que *de el* se puede reescribir como *del* abreviadamente.
- *you're* es una contracción de *you are* ya que *you are* se puede reescribir como *you're* abreviadamente.

La regla anterior se cumple para ambigüedades también, por ejemplo:

- *I'll* es una contracción de *I will* ya que *I will* se puede reescribir como *I'll* abreviadamente.
- *I'll* es una contracción de *I shall* ya que *I shall* se puede reescribir como *I'll* abreviadamente.

Este módulo se encarga de separar una contracción en sus diferentes componentes, utilizando para ello un diccionario externo que especifica cómo se deben descomponer dichas contracciones, de manera que sólo es necesario modificar esta información para adaptar este módulo a otro idioma.

Por ejemplo, la salida correspondiente a la contracción *you're* es *you are*, es decir, *you're* se ha descompuesto en el pronombre *you* y la conjugación verbal *are*.

Debe tenerse en cuenta que una misma contracción puede derivarse en dos o más frases, produciendo ambigüedades que deben resolverse en una fase posterior donde haya mayor información para determinar la separación correcta. Por ejemplo, en el idioma inglés la contracción *I'd* puede referirse indistintamente a *I would*, *I should* o *I had*.

Para que el preprocesador pueda realizar su tarea en esta etapa, podría optarse por dos soluciones:

- Separar las contracciones ambiguas en todas sus posibilidades dejando una marca a cada una de ellas, así, en la fase posterior a quien corresponda eliminar la ambigüedad deberá escoger una de estas alternativas. Si se hace de esta manera, debe tenerse en cuenta que en la etapa de **tokenización**

(posterior al preprocesamiento) se deberá utilizar esta marca para hacer el manejo adecuado de tokens. Por ejemplo, en el caso mencionado de *I'd*, si se opta por esta primer opción, el preprocesador en esta etapa podría generar una salida como `#{I had, I should, I would}#`. El tokenizador debe tener en cuenta entonces esta marca para determinar que está tratando con alternativas de tokens, los cuales son excluyentes (es válido solo uno de ellos).

- Otra opción a tratar en este módulo, es tomar sólo las contracciones que no generen ambigüedades y dejar las contracciones ambiguas sin modificación alguna. Así, *I'd* no se trataría en esta fase pero *You're* sí, requiriendo entonces que el tokenizador trate la contracción *I'd* como un solo token dejando el trabajo de separación y eliminación de ambigüedades para una fase posterior.

3.1.2.2. Implementación del prototipo del módulo de contracciones

Para el prototipo de esta etapa se ha optado por implementar la segunda solución explicada anteriormente, pues implementar la primer solución hace más complejo el desarrollo de la siguiente etapa (tokenización) e implicaría que éste se comportara de diferentes maneras dependiendo del problema que se esté tratando, es decir, tendrían que generarse también marcas para los tokens cuando se traten de alternativas para un único token.

Al igual que el módulo filtro, este módulo es independiente del idioma. En este caso el módulo de contracciones separa las contracciones del idioma inglés, pero si se va a trabajar con otro idioma como fuente basta con escribir en este archivo con las contracciones correspondientes. Esto quiere decir que si hay dos idiomas *a* y *b*, y ninguna de las contracciones del idioma *a* se repite en el idioma *b*, no es necesario entonces reemplazar todo el contenido del archivo *contracciones.txt* y bastará simplemente con agregarle el nuevo contenido.

La estructura de este archivo es como sigue:

Contracción origen	Opciones de separación
Contracción_1	Separacion_11 separacion_12 ... separacion_1i
Contracción_2	Separacion_21 separacion_22 ... separacion_2j
...	...
Contracción_n	Separacion_n1 separacion_n2 ... separacion_nm

Como se puede apreciar hay una línea por cada contracción, en la primer columna se indica la contracción origen, seguido pueden ir tabulaciones o cualquier otro carácter de espaciado y finalmente se indican las diferentes separaciones a que puede dar lugar, separadas por el símbolo *pipe* (`|`). Ejemplos:

isn't	is not
can't	can not
i'll	i will i shall
you'd	you would you had you should
Del	de el
trescientos	tres cientos

En el ejemplo anterior se puede visualizar un archivo de contracciones con cuatro registros del inglés y dos del español.

En la Figura 4 se muestra un ejemplo de un texto antes y después de ser preprocesado por el módulo de contracciones, que utiliza una lista de contracciones predeterminada la cual se encuentra en el Anexo H.

Figura 4 Texto antes y después de expandir contracciones

Antes del Preprocesamiento	Después del Preprocesamiento
Your style will be warmer and truer to your personality if you use contractions like "I'll" and "willn't" and "can't" when they fit comfortably into what you're writing. "I'll be glad to see them if they don't get mad" is less stiff than "I will be glad to see them if they do not get mad." (Read that aloud and hear how stilted it sounds.) There's no rule against such informality—trust your ear and your instincts.	Your style will be warmer and truer to your personality if you use contractions like "I'll" and "will not" and "can not" when they fit comfortably into what you are writing. "I'll be glad to see them if they do not get mad" is less stiff than "I will be glad to see them if they do not get mad." (Read that aloud and hear how stilted it sounds.) there is no rule against such informality-trust your ear and your instincts.

Nótese que la contracción *I'll* no se expandió por el hecho de ser ambigua en esta etapa, pues bien podría ser *I will* o *I shall*.

El prototipo de esta etapa contiene tres funciones:

- Una principal llamada `_expandir_contracciones` que toma el archivo a expandir (en formato `.pff`), crea un archivo del mismo nombre pero con extensión `pef` (*Preprocessing Expanded File*)¹⁰², llama a la función que lee el archivo de contracciones y a la función que reemplaza en el archivo de salida las contracciones por su expansión.

¹⁰² En el presente trabajo, se ha creado una extensión para nombrar el archivo de salida en esta etapa. Esta extensión no es un estándar y el archivo generado aquí es usado temporalmente para crear una interfaz con el módulo de expansión de preprocesamiento dependiente del idioma.

- La función `_leer_contracciones` además de leer el contenido del archivo recibido, deposita su contenido en una lista de contracciones y expansiones.
- La función `_mapear_contracciones` reemplaza cada contracción de la lista leída, por su correspondiente expansión en el archivo de salida. Aquí se tienen en cuenta solamente las contracciones que dan lugar a una y solamente una expansión.

Para el módulo de contracciones es necesario importar `re` para el manejo de expresiones regulares, `environ` para hacer referencia a variables de entorno, `sep` que se refiere al separador que usa el sistema operativo para separar rutas y `Archivo` de `traductor.librerias.IO` para las operaciones de lectura y escritura de archivos.

```
from traductor.librerias.IO import Archivo
from os import environ
from os.path import sep
import re
```

Función principal: administra la interacción entre las otras dos funciones.

```
def _expandir_contracciones(self):
    if not environ.has_key('ARCHIVOS_TRADUCTOR'):
        raise IOError, 'No se encuentra la carpeta de archivos de texto.'
        Defina la variable de entorno ARCHIVOS_TRADUCTOR'
    entrada = Archivo(self.archivo.get_nombreSolo() + '.pff', 'r')
    salida = Archivo(self.archivo.get_nombreSolo() + '.pef')
    textoEntrada = entrada.leer()
    listaContracciones = self._leer_contracciones(
        environ['ARCHIVOS_TRADUCTOR'] + sep +
        'Contracciones.txt')
    textoSalida = self._mapear_contracciones(textoEntrada,
        listaContracciones)
    salida.escribir(textoSalida)
```

Lectura de contracciones: lee el contenido del archivo plano depositándolo en una lista.

```
def _leer_contracciones(self, archivo):
    entrada = Archivo(archivo, 'r')
    listaContracciones = []
    separador = "\W+ \W+"
    expExpansiones = "\W+\W+"
    entrada.leer()
    for i in range(entrada.get_num_lineas()):
        linea = entrada.get_linea()
```

```

linea = linea.replace('\n', '')
if linea != '':
    contraccionExpansion = re.split(separador, linea)
    cadContraccion = contraccionExpansion[0]
    cadExpansiones = contraccionExpansion[1]
    listaExpansiones = re.split(expExpansiones, cadExpansiones)
    elemento = [cadContraccion, listaExpansiones]
    listaContracciones.append(elemento)
return listaContracciones

```

Mapeo de contracciones: por cada una de las contracciones y su correspondiente separación, reemplaza la contracción en el texto. Sólo se tienen en cuenta las contracciones que no son ambiguas (dan lugar a una sola separación). Nótese que para el patrón se usa la bandera `re.I`, esto permite que la búsqueda de la contracción en el texto se haga independientemente de la *capitalización* de la misma.

```

def _mapear_contracciones (self, texto, listaContracciones):
    salida = texto
    for contraccion in listaContracciones:
        if len(contraccion[1]) == 1:
            opciones = ' '.join(contraccion[1])
            patron = re.compile(contraccion[0], re.I)
            salida = patron.sub(opciones, salida)
    return salida

```

3.1.3. Preprocesamiento dependiente del idioma

Aplicado el filtro y el separador de contracciones queda el texto listo para ser tokenizado, pero dependiendo del idioma que se tenga como fuente, pueden hacerse diferentes labores adicionales que faciliten el tratamiento posterior del mismo. Por ejemplo, en el idioma inglés podrían expandirse las contracciones posesivas y dejar las frases con su correcta separación. Una frase como *the boy's hat* debe quedar expandida como *the hat of the boy*¹⁰³.

3.1.3.1. Modelamiento del módulo de procesamiento específico

Si en un idioma I a una cadena s se le aplica una regla R_i obteniendo como resultado la cadena s' , teniendo además que s y s' tienen el mismo significado. Se puede decir entonces que:

¹⁰³ "The Apostrophe". 28 Jan 2006.

<http://owl.english.purdue.edu/handouts/grammar/g_apost.html>.

$s' = R_i(s)$, tal que s' puede sustituir a s sin alterar el significado.

Por ejemplo, sea R_i la regla para contracciones posesivas en el inglés y s la cadena the hat of the boy. De esta manera s' será the boy's hat. Es claro en este ejemplo que la regla R_i se puede aplicar específicamente para el idioma inglés.

Ahora bien se podría entonces definir una regla R_i^{-1} de manera inversa, de tal forma que si se tiene la cadena the boy's hat dé como resultado la cadena the hat of the boy.

$s = R_i^{-1}(s')$, tal que s puede sustituir a s' sin alterar el significado.

Las reglas R_i^{-1} deben definirse con mucho cuidado de tal manera que no den espacio a errores y mucho menos a ambigüedades. Para realizar dichas definiciones debe de igual manera conocerse muy bien el idioma específico en que se está trabajando

3.1.3.2. Implementación del prototipo de procesamiento específico

Hay un problema con el uso de la palabra *one* para referirse a un sustantivo en contexto, por ejemplo, The Colombia's capital is Bogota and Noruega's one is Oslo debería quedar como The capital of Colombia is Bogota and the capital of Noruega is Oslo. El problema es que el preprocesador no cuenta con la información suficiente para hacer este tipo de deducciones, así que su salida será algo como: The capital of Colombia is Bogota and one of Noruega is Oslo. Este problema se podría resolver de la misma manera que se enfrentan las contracciones ambiguas, colocando una marca sobre la palabra *one* cuando esta provenga del tratamiento de una frase posesiva y así se tenga en cuenta esta marca en la tokenización y en fases posteriores, o no separando estas frases cuando la palabra que sigue sea *one*. Una tercera opción podría ser la de remplazar la ocurrencia de *one* después del posesivo por la palabra que haya correspondido a la última separación.

Cualquiera que sea la solución, lo ideal es que se trate el tema de manera completa sin llevar problemas para fases posteriores, ya que al tratarse de una fase que es dependiente del idioma, su implementación puede afectar de manera negativa el comportamiento de los posteriores módulos pues no tienen en cuenta las marcas que aquí se llegasen a generar.

El prototipo desarrollado para esta etapa inicialmente trató el problema de separación de frases posesivas del idioma inglés, basándose en la tercera solución para el problema de la palabra *one*. El módulo está abierto para dar

tratamiento a cualquier idioma, basta con implementar las condiciones especiales que apliquen para cada uno. Para nombrar cada idioma se usa la especificación **ISO 639**¹⁰⁴ con códigos de dos letras, así, el idioma español se representa como *es*, el inglés como *en*, el francés como *fr*, etc. Los nombres de las funciones de cada idioma comienzan por el código del idioma, seguido por un guión bajo y por último el nombre de la función. Así, `en_convertir_posesivos` representa la función `_convertir_posesivos` para el idioma inglés.

En la Figura 5 se muestra un ejemplo de un texto antes y después de ser preprocesado por el módulo de preprocesamiento específico para el inglés.

Como se puede observar la salida del preprocesador para la separación de contracciones posesivas es demasiado pobre, de las frases probadas sólo se aproximó a su correcta separación la última, “*Colombia’s capital is Bogota and Noruega’s one is Oslo*”. Esto se debe a que para hacer la correcta separación de dichas contracciones es necesario contar con más información gramatical y semántica de la frase, así que finalmente se optó por dejar el módulo abierto para cualquier implementación, más no se hizo implementación especial para algún idioma.

Figura 5 Texto antes y después de preprocesar contracciones posesivas

Antes del Preprocesamiento	Después del Preprocesamiento
My mother-in-law’s cooking is atrocious. Steven King’s novels are very popular. You can get anything you want at Arlo and Alice’s restaurant. The Hatfields and the McCoys’ time- share condo is in Florida. Dr. Jeckyl’s and Mr. Hyde’s futures were quite bleak. Mr. Sweeny’s and Mrs. Todd’s barber shops are both on Richmond Row. Colombia’s capital is Bogota and	My cooking of mother-in-law is atrocious. Steven novels of King are very popular. You can get anything you want at Arlo and restaurant of Alice. The Hatfields and the McCoys’ time-share condo is in Florida. Dr. and of Jeckyl Mr. futures of Hyde were quite bleak. Mr. and of Sweeny Mrs. barber of Todd shops are both on Richmond Row. capital of Colombia is Bogota and capital

El módulo tiene entonces una función principal denominada `_procesar_condiciones_especiales` que se encarga de leer el archivo de entrada, crear el archivo de salida y llamar cada una de las funciones que se requieran para cada idioma. Es labor entonces de quien desee hacer uso de este módulo el implementar las funciones para el idioma que use como fuente.

¹⁰⁴ "Codes for the Representation of Names of Languages". 27 Jan 2006. <<http://www.loc.gov/standards/iso639-2/englangn.html>>.

La función principal por cada uno de los idiomas hace llamado a las funciones correspondientes. Si se añade una nueva función, su llamado debe instanciarse en el *if* que corresponda a su idioma, si se agrega un nuevo idioma, debe agregarse una instrucción *elif* al final del *if*.

```
import re

def _preprocesar_condiciones_especiales(self, idioma):
    entrada = Archivo(self.archivo.get_nombreSolo() + '.pef', 'r')
    salida = Archivo(self.archivo.get_nombreSolo() + '.psf')
    if idioma == 'es':
        salida = expandir_posesivos(entrada)
    salida.escribir(entrada.leer())
```

3.2. USO DEL PREPROCESADOR

La interfaz de la clase `preprocesador` es una función llamada `preprocesar`. Para acceder a él solo se debe importar el módulo completo y hacer un llamado a la función principal enviando como parámetro la ruta del archivo `.txt` que se usará como texto fuente.

La interfaz del preprocesador importa `re` para el manejo de expresiones regulares y `Archivo` de `traductor.librerias.IO` para las operaciones de lectura, escritura y borrado de archivos. Finalmente genera el archivo de salida con la misma extensión del archivo de entrada, pero alterando el nombre agregándole la cadena `_pre`, así, para el archivo de entrada `texto.txt`, la salida en esta etapa se llamará `texto_pre.txt`.

```
from traductor.librerias.IO import Archivo
import re

def preprocesar(self):
    if self.archivo == "":
        raise Preprocesador, "Error! Debe definir primero el archivo a preprocesar"

    self._filtrar()
    self._expandir_contracciones()
    self._preprocesar_condiciones_especiales('en')
    entrada = Archivo(self.archivo.get_nombreSolo() + '.psf', 'r')
    salida = Archivo(self.archivo.get_nombreSolo() + '_pre' +
                    self.archivo.get_extension(), 'w')
    salida.escribir(entrada.leer())
    self.archivo.borrado_fisico(self.archivo.get_nombreSolo() + '.psf')
    self.archivo.borrado_fisico(self.archivo.get_nombreSolo() + '.pef')
    self.archivo.borrado_fisico(self.archivo.get_nombreSolo() + '.pff')
```

Habiendo tratado el preprocesamiento, encargado de que el texto de entrada esté libre de espacios en blanco innecesarios, como tabulaciones y espacios dobles, identificar las palabras que han sido separadas por un guión y un salto de línea, además de eliminar las contracciones que no generan ambigüedades.

El paso siguiente es tokenizar el texto que otorga el preprocesamiento, cabe destacar que en ocasiones la tokenización va ligada al preprocesamiento, pero se optó por separar los módulos para tratarlos independientemente.

En el siguiente capítulo se tratara entonces el proceso de tokenización que consiste en identificar cada una de las piezas que conforman un texto.

4. TOKENIZACIÓN

4.1. INTRODUCCIÓN

En el capítulo anterior se trató el tema concerniente al preprocesamiento, que permite que el texto de entrada esté libre espacios en blanco innecesarios, como tabulaciones y espacios dobles, se identifican las palabras que han sido separadas por un guión y un salto de línea, además de eliminar las contracciones que no generan ambigüedades.

El proceso de **tokenización** es el segundo paso realizado en un proceso de TA, generalmente va ligado al preprocesamiento, pero se ha decidido separar los procesos para clarificar los problemas que se presentan en cada una de las etapas y solucionarlos separadamente. En este capítulo se tratarán algunos aspectos teóricos concernientes a este proceso, se explicarán los problemas más comunes y finalmente se presentará un prototipo de tokenizador basado en expresiones regulares que permiten identificar con claridad cuales son los tokens en el texto.

En los **lenguajes artificiales**, como lenguajes de programación, la definición de lo que puede ser considerado un **token** puede ser precisa y definida sin ambigüedades. Por ejemplo para un analizador o *parser* de un compilador debería ser fácil poder determinar qué es una variable, qué es un identificador, qué es un número y en general de qué está compuesto un archivo de entrada. Por otro lado los lenguajes naturales, muestran una amplia variedad y muchas alternativas para decidir sobre lo que puede ser considerado una unidad computacional en un texto, esto se debe a la misma naturaleza de este tipo de lenguajes.

La tokenización consiste en el proceso de identificación de cada una de las piezas que conforman un texto, este proceso debe poder realizarse sin ambigüedad. Aunque en primera instancia puede parecer sencillo, no resulta así, pues existen muchas consideraciones que deben tenerse en cuenta.

4.2. ASPECTOS TEÓRICOS

4.2.1. Consideraciones preliminares

A continuación se harán algunas definiciones referentes al proceso de tokenización.

4.2.1.1. El token como parte fundamental de un texto¹⁰⁵

Generalmente en un escrito, una palabra es una pieza de texto, la dificultad radica en la manera cómo se deben representar las palabras y asociar esta información para que sea manejada por un computador. Habría que definir claramente lo qué es una palabra. No se puede simplemente decir que una palabra es una cadena de caracteres con un espacio antes y después, sin duda el tratamiento es un poco más complejo.

Si se trata de tokenizar teniendo en cuenta la definición de la separación de tokens por espacios, ocurren además otra serie de problemas que aunque mínimos no dejan de producir errores en la traducción. Por ejemplo, asumiendo que el archivo de origen se encuentra compuesto por varias líneas, entonces todas las palabras de comienzo de una nueva línea no cumplirán la condición de estar precedidas por un espacio en blanco (a no ser que al carácter de nueva línea se le de un tratamiento similar al del espacio en blanco). Un segundo problema, de acuerdo a este criterio es que los signos de puntuación formarían parte de la palabra que la precede inmediatamente. Esto dificultaría entonces la identificación de los límites de una frase.

Un reto levemente diferente se presenta a continuación con los siguientes ejemplos:

This is a alpha-galactosyl-1,4-beta-galactosyl-specific adhesin.

The corresponding free cortisol fractions in these sera were 4.53 +/- 0.15% and 8.16 +/- 0.23%, respectively.

En estos casos, se encuentran términos que son poco probables de ser encontrados en algún léxico inglés de uso frecuente. Por otra parte, no se tendrán resultados al intentar analizar sintácticamente estas secuencias usando una gramática estándar del inglés. Ahora para algunos usos, se quisiera experimentar con expresiones tales como: *alpha-galactosyl-1,4-beta-galactosyl-specific adhesin* y *4.53 +/- 0.15% and 8.16 +/- 0.23%, respectively* para presentarlas como átomos no analizables al analizador. Es decir, se desea tratarlas como palabras solas para propósitos del proceso siguiente. El resultado es que, incluso si se atiende al texto en inglés, lo que se quiere expresar y las palabras pueden depender mucho del contexto.

¹⁰⁵ Este apartado es una adaptación en español tomada de la documentación de NLTK (Natural Language ToolKit). "Tokens: the building blocks of text" <http://nltk.sourceforge.net/tutorial/tokenization/nochunks.html>

4.2.1.2. Comparación entre token y tipo.

Para comprender la comparación entre estos dos términos se hará referencia al siguiente párrafo:

El perro y el gato son animales domésticos, se pueden tener en casa a diferencia de otros animales como el león y el tigre que son considerados salvajes.

Obsérvese que si se cuenta la cantidad de palabras, el resultado sería 28 palabras, este cálculo consiste en contar todas las palabras incluyendo las ocurrencias múltiples, por ejemplo, la palabra *el* se encuentra repetida cuatro veces, y la palabra *animales* se encuentra dos veces. La discusión se centra en identificar si la palabra *el* se encuentra cuatro veces o simplemente una. En el párrafo de ejemplo se afirmaría que esta palabra son cuatro **tokens** pero un solo **tipo**, es decir, un token de la palabra es algo que existe en tiempo y espacio.

4.2.2. El proceso de tokenización

Muchas de las tareas del procesamiento de lenguaje natural involucran análisis de textos de diferentes tamaños, que van desde oraciones simples hasta cuerpos de texto enormes.

Normalmente en el proceso de **tokenización** se tiene como entrada un archivo plano de caracteres y se obtiene como salida dicho texto en una lista separado adecuadamente, identificando de una manera clara los tokens.

Las expresiones regulares pueden explorar un texto y separar las palabras, indicando los caracteres que pueden ser incluidos para que las palabras puedan ser reconocidas. La aplicación de estas expresiones regulares se observará más adelante en la implementación del prototipo de tokenizador.

El tokenizador se encarga de solucionar diversos problemas que en la mayoría de lenguajes son similares, los problemas más comunes son los que se presentan a continuación.

4.2.2.1. Combinaciones de letras y números

Este tipo de combinaciones es muy especial, pues en ningún diccionario aparecen estas palabras, lo cual no significa necesariamente que no existan o que estén escritas incorrectamente. Ejemplos: UB40, Windows98, 350AC

4.2.2.2. Guiones y signos

Se debe tener en cuenta que ciertas palabras en algunos idiomas cuenta con separación mediante signos como el guión, que comúnmente es más utilizado, el signo @ el cual se usa para escritura de correos electrónicos, además de otros caracteres que se emplean en ocasiones. Ejemplos: Zig-Zag, B-49.

4.2.2.3. Caracteres de puntuación

La puntuación de los textos escritos, con la que se pretende reproducir la entonación de la lengua oral, constituye un capítulo importante dentro de la ortografía de cualquier idioma. De ella depende en gran parte la correcta expresión y comprensión de los mensajes escritos. La puntuación organiza el discurso y sus diferentes elementos y permite evitar la ambigüedad en textos que, sin su empleo, podrían tener interpretaciones diferentes. Este tipo de caracteres se utilizan en los lenguajes para agrupar, hacer énfasis, realizar referencias y separar las oraciones. Entre ellos están:

- Punto .
- Coma ,
- Guión -
- Dos puntos :
- Punto y coma ;
- Puntos suspensivos ...
- Corchetes []
- Paréntesis ()
- Comillas ' ', “ ”, « »
- Signos de interrogación ¿ ?
- Signos de exclamación ¡ !

4.2.2.4. Palabras compuestas

Este tipo de problema puede ser abordado en la etapa de tokenización o ser tratado en una fase posterior. Si el problema es tratado en la tokenización, es necesario que se puedan reconocer las palabras compuestas como un solo token. Ejemplos: Santa María, New York.

4.2.2.5. Mayúsculas y minúsculas

Se debe tener en cuenta si el tokenizador es o no sensitivo a mayúsculas y minúsculas ya que se puede presentar el caso en el que los textos están escritos sólo en mayúsculas, sólo en minúsculas o en una mezcla de ambas, en tal caso

es necesario que el tokenizador pueda manejar este inconveniente. Ejemplos: Motor y motor, IVA e iva.

4.2.2.6. Acentos y tildes

Se refiere a aquellos idiomas en los que se utilizan acentuaciones especiales en algunas palabras, como en el castellano que es utilizada la tilde. Ejemplos: en español solo y sólo.

4.2.2.7. Sentidos de escritura inversa

Algunos idiomas no manejan espacios para separar las palabras, tal como el chino y el japonés; en este caso la tokenización tiene que operar de una manera distinta para lograr su propósito.

4.2.2.8. Múltiples alfabetos

En idiomas como el japonés en el cual hay tres alfabetos: caracteres chinos o *kanjis*, alfabeto silábico de *hiragana* y también *katakana*. Un texto en japonés, utiliza los tres al mismo tiempo. Además, comúnmente también se usa el alfabeto romano.

4.2.2.9. Significado de los números

Los números tienen un comportamiento muy particular y en el momento de la tokenización deben ser tratados adecuadamente. Es así como, un número puede significar un valor monetario, un año, una referencia, una cantidad, entre otros.

4.2.2.10. Formato de escritura de fecha y hora

Las fechas pueden ser escritas de diferentes maneras, y sin importar cual sea su escritura, la interpretación es la misma, en ocasiones se escribe con separadores tales como un *slash* o un guión, y en otras simplemente se escribe la fecha. Ejemplos: 05/09/2005 sería un token completo, 05-09-2005 también sería un token completo y 5 de Agosto de 2005 quedaría de la siguiente forma: '5' 'de' 'Agosto' 'de' '2005'. En el caso de la hora: 10:30 a.m., 10:30 A.M., 10:30 p.m., 10:30 P.M., 10:30, en los cinco casos se tomarían como un solo token toda la expresión.

4.2.2.11. Abreviaciones

Las abreviaciones deben tener un tratamiento especial, ya que estas no necesariamente indican final de una frase por terminar o contener un punto, deben

ser manejadas con cuidado para identificar de una manera adecuada cuando un punto pertenece o no a una abreviación.

Los anteriores problemas se presentan en algunos lenguajes, separar las frases en palabras también depende de interpretación y es en este aspecto donde se pueden encontrar ambigüedades.

4.2.3. Representación de tokens¹⁰⁶

Cuando se almacena un texto en una máquina normalmente es representado mediante una cadena de caracteres. Es decir, a través de un archivo de caracteres. En este tipo de representación, cada palabra es una cadena de caracteres, las frases, son así mismo cadenas de caracteres y en realidad la totalidad del texto es por sí mismo una gran cadena de caracteres. Adicional a esto, se debe tener en cuenta que los caracteres en una cadena no son solamente caracteres alfanuméricos. Las cadenas de caracteres también pueden incluir caracteres especiales que representan espacios en blanco, tabuladores y saltos de línea.

El proceso de cómputo se realiza más a nivel de caracteres. En la compilación de un lenguaje de programación, por ejemplo, el compilador espera que su entrada sea una secuencia de tokens, los cuales sabe reconocer; por ejemplo, las clases de identificadores, constantes, cadenas y números. Análogo, un analizador esperará que su entrada sea una secuencia de tokens en vez de una secuencia de caracteres individuales. Es mas sencillo que el tokenizador busque en un texto las localizaciones en la cadena de caracteres que contienen el espacio en blanco (espacio, tabulador, o nueva línea) o ciertos signos de puntuación, y romper la secuencia en tokens en esos puntos. Por ejemplo, supóngase que se tiene un archivo que contiene las siguientes dos líneas:

```
The cat climbed  
the tree.
```

Desde el punto de vista del analizador, este archivo es una simple cadena de caracteres.

```
'The cat climbed\n the tree.'
```

¹⁰⁶ Este apartado es una adaptación en español tomada de la documentación de NLTK (Natural Language ToolKit). Representing tokens. <http://nltk.sourceforge.net/tutorial/tokenization/nochunks.html>

Nótese que se utilizan comillas simples para delimitar la cadena de caracteres, guión bajo para representar el espacio en blanco, y “\n” para representar una nueva línea.

Como se acaba de precisar, **tokenizar** este texto para ser tomado por el analizador, se requiere indicar explícitamente qué subcadenas son palabras. Una manera conveniente de hacer esto en *Python* es partir la cadena en una lista de palabras, donde cada palabra es una cadena. En Python, las listas se imprimen como una serie de objetos (en este caso, cadenas), que son encerrados por los corchetes y separadas por comas:

```
>>> words = ['the', 'cat', 'climbed', 'the', 'tree']
>>> words
['the', 'cat', 'climbed', 'the', 'tree']
```

Para resumir, se acaba de ilustrar cómo, de la manera más simple, la **tokenización** de un texto puede ser realizada convirtiendo la cadena que representa el texto en una lista de las cadenas, cada uno de las cuales corresponde a una palabra.

4.3. MODELAMIENTO DEL TOKENIZADOR

Como se ha definido previamente los tokens son conformados por la asociación de diferentes caracteres los cuales cumplen unas reglas específicas. Por ejemplo las fechas, los números que representan cantidades monetarias o una palabra sencilla. Para cada una de este tipo asociación entre caracteres se pueden definir reglas que representan la manera en como se constituyen. Dichas reglas se pueden expresar a través de expresiones regulares y en algunos casos son dependientes del idioma (aunque en la mayoría de los idiomas son similares). De igual forma deben ser definidas de manera jerárquica de tal manera que un token con una regla compleja no sea interpretado como varios tokens expresados a través de reglas sencillas. Por ejemplo el token zig-zag debe ser identificado como un token compuesto por dos palabras separadas por un guión y no como tres tokens diferentes (palabra, guión, palabra). De esta manera la regla para identificar tokens compuestos por palabras separados por guiones debe aplicarse antes de aplicar el de palabras simples.

Debe identificarse claramente el orden jerárquico para cada una de las reglas de conformación de tokens.

A continuación se presenta la definición de expresiones regulares y la jerarquía que se debe aplicar para tokenizar correctamente el idioma inglés:

- **Fechas:** Para identificar las fechas, la expresión regular debe poder soportar los diferentes formatos en que estas pueden ser representadas (dd/mm/AAAA, AAAA/mm/dd, mm/dd/AAAA, donde cada letra es representada por el patrón \d que denota un dígito) e igualmente su separador (/ o -). Esto se logra definiéndola mediante :

```
fecha = ' ( (\d{1,4}[/-]){2}\d{1,4}) '
```

- **Horas:** Para identificar la hora, la expresión regular podrá soportar los siguientes formatos: (HH:MM m, HH:MM M, HH:MM a.m., HH:MM A.M., HH:MM P.M., HH:MM p.m.).

```
hora = ' (\d{1,2}:\d{1,2}\s*(m|M|a.m.|p.m.|A.M.|P.M.)?) '
```

- **Números:** Para la identificación de números de representación decimal la expresión regular debe hacer cumplir la condición de que un número real está representado por muchos dígitos seguidos de un punto y este a su vez seguido de por lo menos un dígito (para que se cumpla la condición de ser un numero expresado en decimales), esta condición permite abarcar números como: 3.1416, 0.15, .19, etc.
- **Cantidad de porcentaje:** Aunque en principio no parece muy útil realizar la definición se puede definir que una cantidad que expresa un porcentaje es un número seguido del signo '%'. La expresión regular `numeroPorc`, soporta los números y cantidades que expresan porcentaje.

```
numeroPorc = '\d+(\.\d+)*(<?\w)*\s*%?'
```

- **Valores Monetarios:** Una expresión regular que identifica valores monetarios inicia reconociendo el signo de moneda, posteriormente permite de uno a tres dígitos, seguido a esto, opcionalmente se pueden presentar un punto y tres dígitos, estos últimos obligatoriamente, esto se puede repetir cuantas veces sea necesario y con ello se logra la separación de miles, millones, etc. Y finalmente, de manera opcional también, se puede terminar la cadena con una coma seguida de dos dígitos que marcarían los centavos.

```
moneda = ' (\$?\d{1,3}(\.\d{3})*(\,\d{2})?) '
```

- **Palabras:** Como requisito primordial el tokenizador debe ser capaz de reconocer palabras, independientemente de si son palabras que se encuentran en mayúsculas o minúsculas, con la siguiente expresión regular

se pueden identificar todas aquellas cadenas que cumplan la condición de ser uno o más caracteres alfanuméricos continuos.

```
palabra = '(\w+([-_\.\'"]\w+)*).'
```

- **Signos de Puntuación:** Se definen la totalidad de signos de puntuación.

```
puntua = '(\. {3})|[\.,-:;\[\]\(\)\\"'¿\?!¡!])'
```

Finalmente si hay algún tipo de token que no pueda ser identificado por las expresiones regulares definidas se puede hacer uso de excepciones tal que dichas sean tenidas en cuenta antes que las definiciones anteriores. Dichas excepciones pueden encontrarse en un archivo o en una base de datos. Entre las excepciones más comunes se pueden tener en cuenta las palabras compuestas, palabras de orden complejo y las abreviaciones.

4.4. IMPLEMENTACIÓN DEL TOKENIZADOR

En el momento de implementar un tokenizador es importante hacer uso de una herramienta que pueda realizar reconocimiento de patrones en un texto, una buena opción puede ser el *LEX*¹⁰⁷, sugerido por *Gregory Grefenstette*¹⁰⁸, de igual manera es muy importante que el lenguaje a ser usado en la implementación del tokenizador posea un buen manejo de cadenas de caracteres y además posea soporte para manejar expresiones regulares, que en el problema a ser abordado son imprescindibles.

A continuación se presenta un modelo de tokenizador basado en expresiones regulares haciendo uso de la herramienta *NLTK LITE*¹⁰⁹, el lenguaje de programación utilizado es *Python* que soluciona la gran parte de los problemas mencionados anteriormente.

El siguiente es el contenido del archivo `Tokenizador.py`, el cual es el código fuente del tokenizador realizado.

```
from nltk_lite.tokenize import regexp
from traductor.librerias.IO import Archivo
from os import environ
```

¹⁰⁷ LEX. Es un generador de análisis léxico.

¹⁰⁸ GREFENSTETTE, Gregory G. G. and TAPANAINEN, Pasi P. T. 1994. What is a Word, What is a sentence? Problems of Tokenization.

¹⁰⁹ Para mayor información de cómo usar e instalar NLTK LITE <http://nltk.sourceforge.net/>

```

from os.path import sep
import re

class Tokenizador:
    def __init__(self, archivo=""):
        if archivo != "":
            self.archivo = Archivo(archivo, "r")
        else:
            self.archivo = Archivo()
        self.tokens = []

    def set_archivo(self, archivo=""):
        if archivo != "":
            self.archivo.set_nombre_modulo(archivo, "r")
        self.tokens = []

    def tokenizar(self):
        if not environ.has_key('ARCHIVOS_TRADUCTOR'):
            raise IOError, 'No se encuentra la carpeta de archivos de texto.
                            Defina la variable de entorno ARCHIVOS_TRADUCTOR'
        arcExcepciones = Archivo(environ['ARCHIVOS_TRADUCTOR'] + sep +
                                'Excepciones.txt', 'r')
        arcExcepciones.leer()
        texto = self._marcar_excepciones(arcExcepciones.get_lineas())
        excep = '#EXCEP#.*?#EXCEP#'
        fecha = '((\d{1,4}[/-]){2}\d{1,4})'
        hora = '(\d{1,2}:\d{1,2}\s*(m|M|a.m. |p.m. |A.M. |P.M.)?)'
        numeroPorc = '\d+(\.\d+)*(<?\w)*\s*%?'
        moneda = '(\$?\d{1,3}(\.\d{3})*(\,\d{2})?)'
        palabra = '\w+([-_\.\']\w+)*'
        punctua = '((\.{3})|[\.,-:;\[\]\(\)\\"¿?;!])'
        patrones = excep + '|' + fecha + '|' + hora + '|' + numeroPorc +
                    '|' + moneda + '|' + palabra + '|' + punctua
        self.tokens = list(regexpat(texto, patrones))
        self.tokens = map(self._desmarcar_excepciones, self.tokens)
        self.tokens = map(lambda x: x.strip(), self.tokens)
        return self.tokens

    def _marcar_excepciones(self, listaExcepciones):
        salida = self.archivo.leer()
        for excepcion in listaExcepciones:
            if (excepcion != ''):
                patron = re.compile('((^)|\s)+' + excepcion.replace('.', '\. '),
                                    re.I)
                salida = patron.sub('#EXCEP#' + excepcion + '#EXCEP#', salida)
        return salida

    def _desmarcar_excepciones(self, cadena):
        excepcion = re.split('#EXCEP#', cadena)
        if (len(excepcion) == 3):
            return excepcion[1]
        return (cadena)

```

```
def get_tokens(self):  
    return self.tokens
```

A continuación se ofrece una explicación de la manera en que el programa citado anteriormente resuelve los problemas de tokenización antes expuestos.

Ya se había indicado anteriormente que para la implementación del tokenizador se hizo uso de la herramienta *NLTK LITE*. Para utilizar el módulo de tokenización del *NLTK LITE* se deben importar las librerías asociadas a éste. Igualmente se importa la librería *re* y *regex* para el manejo de las expresiones regulares, *environ* para hacer referencia a variables de entorno, *sep* que se refiere al separador que usa el sistema operativo para separar rutas y *Archivo* de *traductor.librerias.IO* para las operaciones de lectura y escritura de archivos.

```
from nltk_lite.tokenize import regex  
from traductor.librerias.IO import Archivo  
from os import environ  
from os.path import sep  
import re
```

Una de las ventajas que ofrece *Python* es que permite trabajar con expresiones regulares, *NLTK LITE* al estar desarrollado en *Python* hereda esta característica que simplifica el trabajo en cuanto al manejo de cadenas de caracteres. La implementación presentada anteriormente hace un amplio uso de las expresiones regulares de *Python*, las cuales son ampliamente explicadas en la documentación de dicho lenguaje¹¹⁰.

El prototipo implementado es orientado a objetos, por lo cual se debe definir una clase para el proceso de tokenización, esta clase recibe el nombre de *Tokenizador*.

```
class Tokenizador:
```

Esta clase tiene su constructor, que recibe como parámetro el nombre del archivo a tokenizar.

```
def __init__(self, archivo=""):  
    if archivo != "":  
        self.archivo = Archivo(archivo, "r")  
    else:
```

¹¹⁰ Página oficial de documentación de Python: <http://www.python.org/doc/>

```
self.archivo = Archivo()  
self.tokens = []
```

Además tiene una función para cambiar el nombre del archivo a tokenizar.

```
def set_archivo(self, archivo=""):  
    if archivo != "":  
        self.archivo.set_nombre_modos(archivo, "r")  
        self.tokens = []
```

La función principal detecta todos los tokens del texto y los organiza en una lista.

```
def tokenizar(self):
```

Se debe verificar que la variable de entorno *ARCHIVOS_TRADUCTOR* exista.

```
if not environ.has_key('ARCHIVOS_TRADUCTOR'):  
    raise IOError, 'No se encuentra la carpeta de archivos de texto.  
Defina la variable de entorno ARCHIVOS_TRADUCTOR'
```

Para solucionar el problema de las abreviaciones se utilizará la sugerencia realizada en el paper *What is a word, What is a sentence? Problems of tokenization*¹¹¹, definiendo en un archivo de texto una lista de abreviaciones y además palabras especiales que el tokenizador no sea capaz de reconocer, así basta simplemente con adicionar en el archivo de excepciones dichas palabras y la tokenización será exitosa, esto permite que tokens compuestos de dos palabras como *New York* sean tratados como un token. En adelante el conjunto de abreviaciones y palabras especiales se llamara *excepciones*. Tanto las abreviaciones como las palabras especiales son entradas una en cada línea. Se debe cargar entonces el archivo de excepciones, para esto se utiliza la variable `arcExcepciones`.

```
arcExcepciones = Archivo(environ['ARCHIVOS_TRADUCTOR'] + sep +  
                          'Excepciones.txt', 'r')  
arcExcepciones.leer()
```

Antes de evaluar el texto, se deben marcar las excepciones, para esto se utiliza la función `_marcar_excepciones`, el resultado de esta función se almacena nuevamente en la variable `texto`.

```
texto = self._marcar_excepciones(arcExcepciones.get_lineas())
```

¹¹¹ GREFENSTETTE, Gregory G. G. and TAPANAINEN, Pasi P. T. 1994. What is a Word, What is a sentence? Problems of Tokenization. Op. Cit.

Se define la expresión regular para manejar las excepciones, se usa una etiqueta al principio y al final de la excepción `#EXCEP#`. Esta expresión regular permite encontrar en el texto cualquier pieza de texto encerrada entre la etiqueta antes mencionada, esta marcación se realizó anteriormente con la función `_marcar_excepciones`.

```
excep = '#EXCEP#.*?#EXCEP#'
```

Se definen cada uno de los patrones de para los tokens.

```
fecha = '((\d{1,4}[/-]){2}\d{1,4})'
hora = '(\d{1,2}:\d{1,2}\s*(m|M|a.m.|p.m.|A.M.|P.M.))? '
numeroPorc = '\d+(\.\d+)*(<?\w)*\s*%'
moneda = '(\$?\d{1,3}(\.\d{3})*(\,\d{2}))?'
palabra = '\w+([-_,\.\']\w+)*'
puntua = '(\.{3})|[\.,-:;\[\]\(\)\\"¿?;!;]|'
patrones = excep + '|' + fecha + '|' + hora + '|' + numeroPorc +
            '|' + moneda + '|' + palabra + '|' + puntua
```

Se tokeniza el texto de acuerdo a los patrones definidos.

```
self.tokens = list(regexp(texto, patrones))
```

Se aplica un mapeo a la lista de tokens, aplicando la función `desmarcar_excepciones` a la lista, esto para eliminar la marca que tienen las excepciones `'#EXCEP#'`.

```
self.tokens = map(self._desmarcar_excepciones, self.tokens)
```

Finalmente se eliminan los espacios en blanco redundantes que generan las expresiones regulares y se retorna la lista de tokens.

```
self.tokens = map(lambda x: x.strip(), self.tokens)
return self.tokens
```

La función `_marcar_excepciones` recibe el texto y la lista de excepciones y retorna todas las excepciones encerradas por la etiqueta `#EXCEP#`, y resuelve la sensibilidad entre mayúsculas y minúsculas.

```
def _marcar_excepciones(self, listaExcepciones):
    salida = self.archivo.leer()
    for excepcion in listaExcepciones:
        if (excepcion != ''):
            patron = re.compile('(^|s)' + excepcion.replace('.', '\. '),
                                re.I)
            salida = patron.sub('#EXCEP#' + excepcion + '#EXCEP#', salida)
```

```
return salida
```

La función `_desmarcar_excepciones` elimina las etiquetas `#EXCEP#`, como cada excepción esta conformada por tres elementos (`#EXCEP#+excepcion+#EXCEP#`), se retorna únicamente el segundo elemento, que para *Python* es el elemento `excepcion[1]`.

```
def _desmarcar_excepciones(self, cadena):  
    excepcion = re.split('#EXCEP#', cadena)  
    if (len(excepcion) == 3):  
        return excepcion[1]  
    return (cadena)
```

Finalmente se encuentra la función `get_tokens` que sirve para obtener la lista de tokens.

```
def get_tokens(self):  
    return self.tokens
```

A continuación se muestra el resultado de aplicar en un texto de entrada el tokenizador explicado anteriormente. De esta manera, para el siguiente texto de entrada:

This is a word
UB40 Windows98 350AC
alpha-galactosyl-1,4-beta-galactosyl-specific Zig-Zag
Finland's capital
\$2.500 \$1.234.456.450,23
2.12123123 12123123.00000 25%
05/09/2005 5 de Agosto de 2005 05-09-2005
13:10 5:15 a.m. 5:15 P.M.
Mr. Sebastian Marin, Mr. Jose Ferney Franco and Mr. Cesar Fredy Gil develop a tesis using the
dotproject developer
- ? () ,
New York

Se obtiene la siguiente salida:

```
['This', 'is', 'a', 'word', 'UB40', 'Windows98', '350', 'AC', 'alpha-galactosyl-1,4-beta-galactosyl-  
specific', 'Zig-Zag', 'Finland's', 'capital', '$2.500', '$1.234.456.450,23', '2.12123123 ',  
'12123123.00000 ', '25%', '05/09/2005', '5 ', 'de', 'Agosto', 'de', '2005 ', '05-09-2005', '13:10 ',  
'5:15 a.m.', '5:15 P.M.', '.', 'mr.', 'Sebastian', 'Marin', ',', 'mr.', 'Jose', 'Ferney', 'Franco', 'and',  
'mr.', 'Cesar', 'Fredy', 'Gil', 'develop', 'a', 'tesis', 'using', 'the', 'dotproject', 'developer', '-', '?', '(',  
)', ',', 'New York']
```

Se puede observar como la mayoría de los problemas que se pueden presentar en la tokenización para el idioma inglés se resuelven con este prototipo. El

tokenizador contempla palabras, palabras con números, números, palabras separadas por guiones, palabras con contracciones, valores monetarios, porcentajes, fechas, horas, abreviaciones, nombres propios, signos de puntuación y palabras compuestas. Adicionalmente el manejo de las abreviaciones y las palabras especiales en el archivo de excepciones permite tokenizar cualquier tipo de palabra que las expresiones regulares no puedan contemplar.

El prototipo de tokenizador implementado y su diseño se realizaron con un enfoque que no se encuentra documentado en ninguno de los documentos consultados. Acepta una gran cantidad de patrones, lo cual hace que el dominio de palabras y formatos de escritura sea bastante amplio. Aunque se hizo uso de la librería `tokenize` de *NLTK LITE*, en realidad son las expresiones regulares implementadas las que realizan la tarea primaria de tokenización.

En este capítulo se trató el proceso de tokenización y los problemas más comunes del mismo, en el prototipo implementado se resolvieron los problemas inherentes al idioma inglés y se preparó el texto de entrada separándolo en tokens para ser posible que cada uno de estos sea analizado morfológicamente, el cual es el paso a seguir. En el capítulo siguiente se abordará el tema de los diccionarios, ya que el análisis morfológico se apoya en dichos diccionarios, para etiquetar gramaticalmente cada uno de los tokens que recibe de la etapa de tokenización.

5. DICCIONARIO

5.1. INTRODUCCIÓN

Como se pudo observar en el capítulo del estado del arte de la traducción automática, en el análisis realizado al compendio de software de traducción. Existen 146 combinaciones de idiomas para las que se cuenta con diccionario electrónico, de las cuales 17 involucran el idioma español. El diccionario es, según *Hutchins*¹¹², uno de los componentes más importantes de un sistema de traducción automática, en mayor parte se debe a la cantidad de información que puede ser proporcionada por éste. De igual manera, porque es a través de su uso que se realiza el intercambio de significados de una palabra entre un idioma y otro. Es por esto que el diseño de un diccionario de traducción automática debe realizarse con mucho cuidado teniendo en cuenta el incluir información que será necesaria en cada una de las etapas de la traducción. La información sintáctica y semántica son el ejemplo más claro de esto.

Debe agregarse que para diferentes tipos de motores de TA obviamente tendrán diferentes requerimientos en cuanto al contenido y la forma del diccionario. Por ejemplo un diccionario de un sistema interlingual no necesita tener información sobre la forma de traducir cada entrada, todo lo que necesita es asociar las palabras con el apropiado concepto interlingual. En contraste, los sistemas de transferencia tendrán típicamente información acerca del lenguaje de origen y su traducción incluyendo también información sobre el lenguaje destino. Ya que los sistemas de transferencia usan niveles más abstractos de representación, los diccionarios van a tener más información acerca de estos niveles.

5.2. ASPECTOS TEÓRICOS SOBRE LOS DICCIONARIOS

Un **diccionario** es básicamente una lista de palabras ordenadas alfabéticamente con información referente a ciertas propiedades de la palabra. Mientras que las reglas gramaticales definen todas las estructuras lingüísticas posibles en un lenguaje, la descripción individual de las palabras se encuentra en los diccionarios.

5.2.1. Importancia de los diccionarios en la TA

Hay un gran número de razones que manifiestan la importancia de los diccionarios, pero la más importante para que los diccionarios existan en los

¹¹² HUTCHINS, John. Lingüista inglés especializado en traducción automática. Op. Cit.

sistemas de TA es su diversidad en términos de formatos, cobertura, nivel de detalle y un preciso formalismo para la descripción léxica. Además de esto se pueden considerar las siguientes:

- Los diccionarios son los componentes más grandes de un sistema de TA en términos de la cantidad de información que tienen. Normalmente son más que una lista simple de palabras (y así debería ser para garantizar un buen desempeño del sistema), por lo tanto al construir un sistema de TA los diccionarios usualmente son los componentes más dispendiosos de construir.
- Más que ningún otro componente, el tamaño y calidad del diccionario limita el alcance y cobertura del sistema y la calidad de traducción que puede ser esperada.
- Los diccionarios son la parte del sistema en la cual el usuario final puede tener la capacidad de contribuir a la mejora del sistema. De hecho, un usuario final espera poder hacer algunas adiciones al diccionario para hacer que el sistema sea realmente útil. Mientras que los proveedores raramente hacen posible para los usuarios modificar cualquier otro componente, ellos normalmente esperan que los usuarios hagan adiciones a los diccionarios. Es por esto, que desde el punto de vista del usuario, un entendimiento básico de la estructura del diccionario y la capacidad que éste tenga para describir palabras es muy importante.

5.2.2. Diccionarios de papel

Los diccionarios de papel generalmente son colecciones de entradas con unas características determinadas. Es importante tener en cuenta la gran cantidad de información de este tipo que se necesita teniendo presente que un diccionario de tan solo 20.000 entradas es considerado pequeño. De hecho, ningún diccionario puede decirse que está completo, no sólo porque normalmente los diccionarios generalmente se restringen a ellos mismos suprimiendo por lo general algunas palabras, sino también porque constantemente nuevas palabras son acuñadas, usadas en otros sentidos y transformadas a través de procesos morfológicos.

5.2.3. Tipos de información de las palabras

Aunque alguna de la información que se haya típicamente en un diccionario de papel tiene un valor limitado en los sistemas de TA (por ejemplo información sobre la pronunciación la cual es útil para otro tipo de sistemas), en general la calidad y detalle de la información que se necesita para un sistema de TA es por lo menos

igual a la que normalmente se hallaría en un diccionario de papel. De esta manera, es recomendable incluir como mínimo la información que estos contienen.

Además sería ideal poder incluir información sobre el ambiente gramatical de una palabra, cabe anotar que el proceso de traducción se dificulta en la medida en que se incluya este tipo de información en el diccionario. Las dos clases de información sobre el ambiente gramatical se tratan a continuación.

5.2.3.1. Información de subcategorización

La información de subcategorización indica los ambientes sintácticos en los cuales se puede presentar una palabra. Típicamente la información de subcategorización es la información que indica que la palabra en inglés *die* (morir) es un verbo intransitivo ya que solo necesita un sujeto para una frase gramaticalmente correcta. La clase de palabra se expresa en el diccionario a través de una categoría, en este caso podría ser con la letra *i*.

Los verbos no son los únicos que pueden ser subcategorizados. En los sustantivos y adjetivos derivados de los verbos se observa el mismo comportamiento. Un diccionario de inglés adecuado podría reconocer por lo menos veinte diferentes subcategorizaciones de verbos y un número similar de adjetivos y sustantivos.

5.2.3.2. Restricciones selectivas

Las restricciones selectivas describen las propiedades semánticas del ambiente. La información de subcategorización indica que, por ejemplo, el verbo *abotonar* en español se aplica a un sustantivo. De hecho, se puede saber mucho más acerca del verbo. El objeto, o en términos semánticos, el paciente, del verbo tiene que ser algo que se pueda abotonar, tal como una pieza de una prenda de vestir y que el sujeto (o más precisamente el agente) del verbo es normalmente animado. Tal información se refiere comúnmente a las restricciones selectivas. Este tipo de información está implícita en la parte inicial de cada entrada en un diccionario de papel. La información de que el objeto sobre el cual actúa abotonar es inanimado y normalmente una parte de una prenda de vestir debe deducirse de una categoría como *sth* (*some thing*) o algo similar para definirlo, de esta manera en el ejemplo se deberían obtener como posibilidades de dicho objeto *abrigo, chaqueta, camisa*, etc. Sin embargo la categoría no dice de ninguna manera que el sujeto del verbo tiene que ser una entidad animada, ya que ninguna otra entidad puede ejecutar una acción como la de *abotonar*. Se asume que un lector humano puede distinguir esto por si mismo. Este tipo de información debe hacerse explícita para poder ser usada en un sistema de TA. Información inherente básica e información sobre

subcategorización y restricciones selectivas pueden ser representadas expresamente para propósitos de la TA. Esencialmente las entradas en un diccionario de TA serán equivalentes a colecciones de atributos y valor. Por ejemplo se podría tener la siguiente información para el sustantivo *botón*, indicando que su base o raíz es *botón*, que es un sustantivo común, el cual es concreto (es decir que no es abstracto como *felicidad* o *sinceridad*).

5.3. MODELAMIENTO DEL DICCIONARIO

5.3.1. Características esenciales de la estructura del diccionario

La estructura con que debe contar el diccionario debe tener grandes ventajas en cuanto al acceso, interpretación y fácil búsqueda de información de tipo sintáctica y semántica.

Se debe optar por emplear una estructura de fácil uso y sencilla y no una estructura que pueda llegar a consumir muchos recursos.

De igual manera, se debe poder incluir información de subcategorización y restricciones selectivas. En este orden de ideas una Base de Datos, por ejemplo, requeriría de un modelo entidad relación de un complejo diseño.

La estructura de este diccionario debe ser muy semejante a la de los diccionarios de papel, con los cuales los usuarios están normalmente familiarizados. En este mismo orden de ideas el diccionario debe ser insensitivo de tal forma que una palabra pueda sea única sin importar si está escrita en mayúsculas o minúsculas.

Y aunque si bien el diccionario, es uno de los elementos que más información proporciona en el proceso de traducción, debe estar implementado de manera sencilla preferiblemente en pocas líneas de código y de igual forma debe encontrarse muy bien documentado.

Preferiblemente la edición del diccionario debe poder hacerse directamente en un archivo de texto, permitiendo una fácil actualización y el poder cambiar dicho archivo para hacer uso de un diccionario con diferente información. Debe ser similar a trasladar la información incluida en un diccionario de papel, lo cual lo hará intuitivo para el usuario y de excelente escalabilidad.

5.3.2. Estructuras Hash

Teniendo en cuenta lo anterior una buena manera de modelar el diccionario es a través de tablas hash. Según wikipedia¹¹³ una tabla hash se puede definir como:

Una tabla hash o mapa hash es una estructura de datos que asocia llaves o claves con valores. La operación principal que soporta de manera eficiente es la búsqueda: permite el acceso a los elementos (teléfono y dirección, por ejemplo) almacenados a partir de una clave generada (usando el nombre o número de cuenta, por ejemplo). Funciona transformando la clave con una función hash en un hash, un número que la tabla hash utiliza para localizar el valor deseado.

Una tabla *hash* tiene como principal ventaja que el acceso a los datos suele ser muy rápido si se cumplen las siguientes condiciones:

- Una razón de ocupación no muy elevada (a partir del 75% de ocupación se producen demasiadas colisiones y la tabla se vuelve ineficiente).
- Una función resumen que distribuya uniformemente las claves. Si la función está mal diseñada, se producirán muchas colisiones.

Los inconvenientes de las tablas *hash* son:

- Necesidad de ampliar el espacio de la tabla si el volumen de datos almacenados crece. Se trata de una operación costosa.
- Dificultad para recorrer todos los elementos. Se suelen emplear listas para procesar la totalidad de los elementos.
- Desaprovechamiento de la memoria. Si se reserva espacio para todos los posibles elementos, se consume más memoria de la necesaria; se suele resolver reservando espacio únicamente para punteros a los elementos.

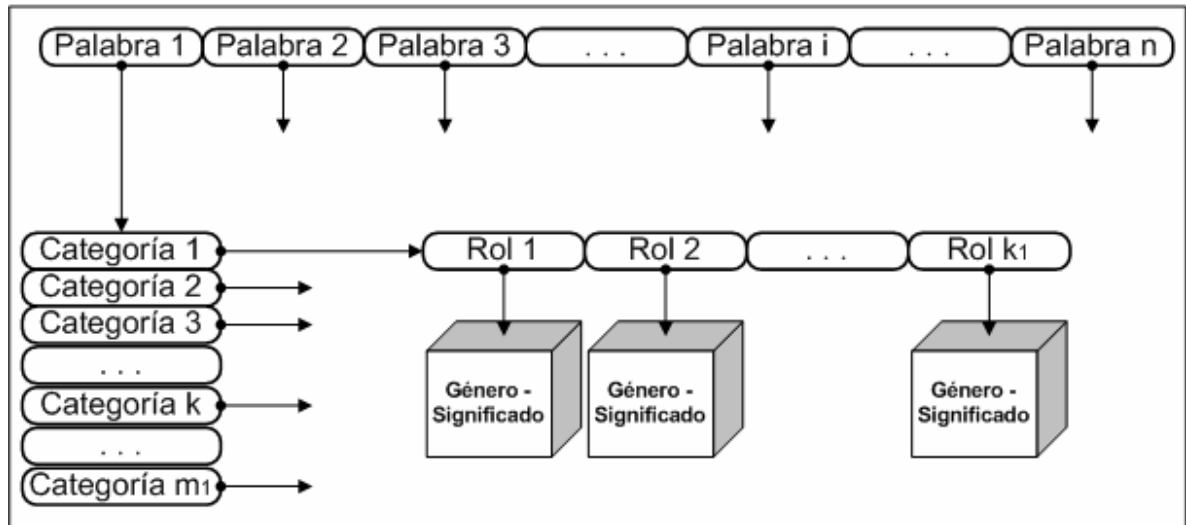
5.3.3. Diagrama del diccionario basado en Hash

De acuerdo a la definición anterior, si se cuenta con un buen algoritmo de hashing es posible tener un rápido acceso a los datos. Si a esto se suma que un diccionario de papel es muy similar a una tabla hash ya que este se encuentra indexado por palabras, las palabras tienen así mismo índices indicando las diferentes categorías de los significados de las palabras y dichas categorías y tienen asociados múltiples roles, por último cada rol tiene un significado y opcionalmente, en caso de ser un sustantivo, dicho significado posee la información de género. De esta manera la estructura puede modelarse como se muestra en la Figura 6:

¹¹³ www.wikipedia.org

Figura 6 Modelamiento estructura del diccionario

Diccionario:



Aunque en un principio el diseño puede parecer complejo, el acceso a los datos es bastante sencillo. Por ejemplo, para realizar la búsqueda de la información referente a una palabra en el diccionario basta con ejecutar una instrucción como las siguientes:

diccionario[palabra]: obteniendo el arreglo asociativo que contiene las diferentes categorías, roles y significados para la palabra dada. Ejemplo:

```
>>> diccionario['play']
{'NN': {'Dep': ['juego '],
        'NOROL': ['juego '],
        'Teat': ['obra']
      },
 'VB': {'Dep': ['jugar '],
        'Mus': ['tocar'],
        'NOROL': ['jugar '],
        'Teat': ['actuar ']
      }
}
```

diccionario[palabra][categoria]: obteniendo el arreglo asociativo que contiene los diferentes roles cada uno con su respectivo significado para la palabra y categoría dadas.

```
>>> diccionario['play']['VB']
{'Dep': ['jugar '],
 'NOROL': ['jugar '],
 'Mus': ['tocar'],
 'Teat': ['actuar ']
}
```

diccionario[palabra][categoria][rol]: obteniendo el significado para la palabra, categoría y rol dados.

```
>>>diccionario['play']['VB']['Dep']  
['jugar ']
```

Es importante de igual forma hacer uso de etiquetas estandarizadas para las diferentes categorías de una palabra. Para esto se aconseja usar el *Brown Corpus Tag Set* (Véase el Anexo F).

5.4. IMPLEMENTACIÓN DEL PROTOTIPO DE DICCIONARIO

5.4.1. Representación de un diccionario en un sistema de TA

La forma más común para realizar esta representación sería un registro de una base de datos, es decir crear un modelo entidad relación que se adapte a las necesidades que se tienen para un diccionario de TA. Aún así el diseño de un diccionario de este tipo es de grandes proporciones y extremadamente complejo. Es por esto que se presenta el siguiente diseño de diccionario basado en la estructura de datos de Python de arreglos asociativos comúnmente conocida como diccionarios¹¹⁴. Las líneas siguientes son la explicación de los diccionarios de Python que se encuentran en su documentación en español¹¹⁵.

A diferencia de las secuencias, que se indexan mediante un rango de números, los diccionarios se indexan mediante claves, que pueden ser de cualquier tipo inmutable. Lo mejor es pensar en un diccionario como en un conjunto desordenado de parejas *clave:valor*, con el requisito de que las claves sean únicas (dentro del mismo diccionario). Una pareja de llaves crea un diccionario vacío: $\{\}$. Si se coloca una lista de parejas *clave:valor* entre las llaves se añaden parejas *clave:valor* iniciales al diccionario. Así es como se presentan los diccionarios en la salida.

Las operaciones principales sobre un diccionario son la de almacenar una valor con una clave dada y la de extraer el valor partiendo de la clave. También se puede eliminar una pareja *clave:valor* con `del`. Si se introduce una clave que ya existe, el valor anterior se olvida. Intentar extraer un valor utilizando una clave no existente provoca un error. El método `keys()` de un objeto de tipo diccionario devuelve todas las claves utilizadas en el diccionario, en orden aleatorio (si se quiere ordenarlas, se aplica el método `sort()` a la lista de claves). Para comprobar si una clave existe en el diccionario, se utiliza el método `has_key()` del diccionario.

De igual manera se incluye un ejemplo que muestra su manera de funcionar.

```
>>> telefonos = {'sebastian': 4098, 'fredy': 4139}
>>> telefonos['john'] = 4127
>>> telefonos
{'fredy': 4139, 'john': 4127, 'sebastian': 4098}
```

¹¹⁴ Aunque para la mayoría de programadores los arreglos asociativos se conocen simplemente como diccionarios en el presente trabajo se hará poco uso de este término, ya que es posible que sea confundido con el término diccionario que hace referencia al diccionario de TA usado en el sistema, así como también al archivo de caracteres en donde se guardarán las palabras. Es importante dejar en claro que, si bien el diccionario de TA del prototipo implementado está basado en arreglos asociativos es una estructura mucho más compleja.

¹¹⁵ Documentación de Python: Tutorial de Python en español". 7 Feb 2006.
<<http://pyspanishdoc.sourceforge.net/>>.

```

>>> telefonos['sebastian']
4098
>>> del telefonos['fredy']
>>> telefonos['mario'] = 4127
>>> telefonos
{'john': 4127, 'mario': 4127, 'sebastian': 4098}
>>> telefonos.keys()
['john', 'mario', 'sebastian']
>>> telefonos.has_key('john')
True

```

Se puede notar la similitud que tiene esta estructura con los diccionarios de papel (de aquí el porque dicha estructura tiene este nombre), además como la facilidad en su uso puede ser ampliamente aprovechada para trabajar en TA.

5.4.2. Formato de entrada y salida

Haciendo uso de los arreglos asociativos en Python se carga el diccionario, la entrada de éste se lee desde un archivo de caracteres que en cada línea tiene una palabra y su correspondiente significado. El formato de cada línea del archivo de diccionario es:

*palabra : categoria1 . [rol1 :] [genero -] significado [; rol2 : [genero -] significado] ...]
[; categoria2 . [rol1:] [genero -] significado [; rol2 : [genero -] significado] ...] ...]*

Entonces, para siguientes palabras la entrada del diccionario sería:

cry: v. llorar; n. llanto
crying: n. llanto; adj. lloroso
device: n. dispositivo
play: v. Dep: jugar + Teat: actuar + Mus: tocar; n. Dep: juego + Teat: obra

Por **categoria** se entiende la **clasificación sintáctica** de la palabra, es decir si la palabra es un verbo (*v*), sustantivo (*n*), adjetivo (*adj*), etc. Por **rol** se entiende la **clasificación semántica** de la palabra, por ejemplo: *Mús* → *Música*, *Zool* → *Zoología*, y por **género** se entiende la clasificación de masculino o femenino para sustantivos, adjetivos, artículos y pronombres, el género va adherido al significado.

Las categorías son obligatorias, los roles y los géneros son opcionales, en caso de no haber ningún rol en un significado de una palabra, el programa que carga la palabra al diccionario la asocia a un rol por defecto llamado *NOROL* con el fin de guardar la estructura del diccionario.

El arreglo asociativo creado tiene el siguiente formato:

```
{'palabra1': {'categorial': {'roll': 'Sig'}, 'categoria2': {'roll': 'Sig'}, ...}, 'palabra2':
{'categorial': {'roll': 'Sig'}, 'categoria2': {'rol': 'Sig'}, ...}, ...}
```

Es importante tener en cuenta que aunque las categorías en el diccionario de texto son las que normalmente se encontrarían en un diccionario de papel, el arreglo asociativo cargado tiene cargadas las categorías con el formato descrito en el estándar *The Brown Corpus Tag-set*, utilizado por NLTK y por *AMALGAM*¹¹⁶ para categorizar (Véase el Anexo F)

Teniendo en cuenta lo anterior, se muestra la estructura del diccionario teniendo únicamente cuatro entradas (*play*, *cry*, *important*, *girl*):

```
>>> diccionario
```

```
{'play': {'VB': {'Dep': ['jugar '], 'NOROL': ['jugar '], 'Mus': ['tocar'], 'Teat': ['actuar ']}}, 'NN':
{'Dep': ['juego '], 'NOROL': ['juego '], 'Teat': ['obra']}}, 'cry': {'VB': {'NOROL': ['llorar']}}, 'NN':
{'NOROL': ['llanto']}}, 'important': {'JJ': {'NOROL': ['importante']}}, 'girl': {'NN': {'Hum': ['f-
niña']}} }
```

De esta manera, para realizar la búsqueda de la información referente a una palabra en el diccionario basta con ejecutar una instrucción como las siguientes:

diccionario[palabra]: obteniendo el arreglo asociativo que contiene las diferentes categorías, roles y significados para la palabra dada. Ejemplo:

```
>>> diccionario['play']
```

```
{'VB': {'Dep': ['jugar '], 'NOROL': ['jugar '], 'Mus': ['tocar'], 'Teat': ['actuar ']}}, 'NN': {'Dep':
['juego '], 'NOROL': ['juego '], 'Teat': ['obra']}}
```

diccionario[palabra][categoria]: obteniendo el arreglo asociativo que contiene los diferentes roles, cada uno con su respectivo significado para la palabra y categoría dadas.

```
>>> diccionario['play']['VB']
```

```
{'Dep': ['jugar '], 'NOROL': ['jugar '], 'Mus': ['tocar'], 'Teat': ['actuar ']}
```

diccionario[palabra][categoria][rol]: obteniendo el significado para la palabra, categoría y rol dados.

```
>>> diccionario['play']['VB']['Dep']
```

```
['jugar ']
```

¹¹⁶ AMALGAM (Automatic Mapping Among Lexico-Grammatical Annotation Models). En español: Mapeo Automático entre Modelos de Anotación Lexico-Gramaticales. Es un proyecto de algoritmos de mapeo de la Universidad de Leeds en Inglaterra.

5.4.3. Explicación del código fuente

Para cargar el diccionario en memoria a partir del archivo de texto se utilizó una función desarrollada en *Python* `_cargar_diccionario` que se encuentra en el archivo `Diccionario.py`.

```
from traductor.librerias.DiccionarioInsensitive import KeyInsensitiveDict
from traductor.librerias.IO import Archivo
from os import remove, environ
from os.path import sep

class Diccionario:
    def __init__(self, archivo=""):
        if not environ.has_key('ARCHIVOS_TRADUCTOR'):
            raise IOError, 'No se encuentra la carpeta de archivos de texto.
                            Defina la variable de entorno ARCHIVOS_TRADUCTOR'
        self.archivo = Archivo(environ['ARCHIVOS_TRADUCTOR'] + sep +
                               archivo, 'r')
        self.diccionario = KeyInsensitiveDict()

    def set_archivo(self, archivo=""):
        self.archivo.set_nombre(environ['ARCHIVOS_TRADUCTOR'] + sep +
                                archivo)
        self.diccionario = KeyInsensitiveDict()

    def cargar(self):
        etiquetasBrown = {'adj pos': 'PP5', 'v': 'VB', 'n': 'NN',
                          'adj': 'JJ', 'pron': 'PPS', 'prep': 'IN',
                          'art': 'AT', 'adv': 'RB', 'conj': 'CC'}
        for i in range(self.archivo.get_num_lineas()):
            lineaArchivo = self.archivo.get_linea()
            palabraSignificado = lineaArchivo.split(': ')
            palabraSignificado = palabraSignificado[0:1] +
                                  [': '.join(palabraSignificado[1:])]
            self.diccionario[palabraSignificado[0]] = KeyInsensitiveDict()
            listaCategoriasSeparadas = palabraSignificado[1].split('; ')
            for categoriaSeparada in listaCategoriasSeparadas:
                categoriaSignificado = categoriaSeparada.split('. ')
                self.diccionario[palabraSignificado[0]]\
                [etiquetasBrown[categoriaSignificado[0]]]
                = KeyInsensitiveDict()
                listaRolesSeparados = categoriaSignificado[1].split('+ ')
                primerRol = ''
                for rolSeparado in listaRolesSeparados:
                    rolSignificado = rolSeparado.split(': ')
                    if ([rolSeparado] == rolSignificado):
                        rol = 'NOROL'
                        indice = 0
                    else:
                        rol = rolSignificado[0]
                        indice = 1
```

```

        if primerRol == '':
            primerRol = rol
        if self.diccionario[palabraSignificado[0]]\
[etiquetasBrown[categoriaSignificado[0]]].has_key(rol):
            self.diccionario[palabraSignificado[0]]\
[etiquetasBrown[categoriaSignificado[0]]][rol]\
= self.diccionario[palabraSignificado[0]]\
[etiquetasBrown[categoriaSignificado[0]]][rol] +
            rolSignificado[indice].split(', ')[0]
        else:
            self.diccionario[palabraSignificado[0]]\
[etiquetasBrown[categoriaSignificado[0]]][rol]\
= rolSignificado[indice].split(', ')[0]
        if not self.diccionario[palabraSignificado[0]]\
[etiquetasBrown[categoriaSignificado[0]]].\
has_key('NOROL'):
            self.diccionario[palabraSignificado[0]]\
[etiquetasBrown[categoriaSignificado[0]]]\
['NOROL'] = self.diccionario[palabraSignificado[0]]\
[etiquetasBrown[categoriaSignificado[0]]]\
[primerRol]

    return self.diccionario

def tiene_palabra(self, palabra):
    if self.diccionario.has_key(palabra):
        return True
    else:
        return False

def tiene_categoria(self, palabra, categoria):
    if self.tiene_palabra(palabra):
        if self.diccionario[palabra].has_key(categoria):
            return True
    return False

def tiene_rol(self, palabra, categoria, rol):
    if self.tiene_categoria(palabra, categoria):
        if self.diccionario[palabra][categoria].has_key(rol):
            return True
    return False

def get_diccionario(self):
    return self.diccionario

def get_palabras(self):
    return self.diccionario.keys()

def get_categorias(self, palabra):
    if self.tiene_palabra(palabra):
        return self.diccionario[palabra].keys()
    else:
        return []

```

```

def get_rol(self, palabra, categoria):
    if self.tiene_categoria(palabra, categoria):
        return self.diccionario[palabra][categoria].keys()
    return []

def get_rol(self, palabra, categoria, rol):
    if self.tiene_rol(palabra, categoria, rol):
        return self.diccionario[palabra][categoria][rol]
    return ""

```

En primer lugar se importa la librería `DiccionarioInsensitive`. Se hace uso de la clase `KeyInsensitiveDict v1.0` desarrollada por *Sami Hangaslammi*¹¹⁷, su finalidad es que se puedan manejar las llaves independientemente de si están en mayúsculas o minúsculas. De igual forma se importa `environ` para hacer referencia a variables de entorno, `sep` que se refiere al separador que usa el sistema operativo para separar rutas y `Archivo` de `traductor.librerias.IO` para las operaciones de lectura y escritura de archivos.

```

from traductor.librerias.DiccionarioInsensitive import KeyInsensitiveDict
from traductor.librerias.IO import Archivo
from os import remove, environ
from os.path import sep

```

Se define la clase `Diccionario` y su constructor, que se encarga de verificar que la variable de entorno se encuentre establecida y que el archivo de texto que contiene el diccionario.

```

class Diccionario:
    def __init__(self, archivo=""):
        if not environ.has_key('ARCHIVOS_TRADUCTOR'):
            raise IOError, 'No se encuentra la carpeta de archivos de texto.
                            Defina la variable de entorno ARCHIVOS_TRADUCTOR'
        self.archivo = Archivo(environ['ARCHIVOS_TRADUCTOR'] + sep +
                               archivo, 'r')
        self.diccionario = KeyInsensitiveDict()

```

Además se define la función `set_archivo` para cambiar el nombre del archivo a cargar como diccionario.

```

def set_archivo(self, archivo=""):
    self.archivo.set_nombre(environ['ARCHIVOS_TRADUCTOR'] + sep +
                             archivo)

```

¹¹⁷ HANGASLAMMI, Sami. <http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/66315>


```
self.diccionario = KeyInsensitiveDict()
```

Se define la función `cargar`. Que se encargara de cargar en memoria el diccionario.

```
def cargar(self):
```

Se define un arreglo asociativo que manejará la correspondencia entre las categorías usadas por el diccionario de texto y las etiquetas estándar del Brown Corpus.

```
etiquetasBrown = {'adj pos': 'PP5', 'v': 'VB', 'n': 'NN',  
                  'adj': 'JJ', 'pron': 'PPS', 'prep': 'IN',  
                  'art': 'AT', 'adv': 'RB', 'conj': 'CC'}
```

Se inicia un ciclo que recorrerá cada línea del archivo. En cada línea se encuentra una palabra y su correspondiente significado. Con este ciclo se pretende cargar en la variable `diccionario` el contenido del archivo, ya que la manipulación de las palabras y significados será mucho más fácil de esta manera.

```
for i in range(self.archivo.get_num_lineas()):
```

Se asigna a una variable temporal una lista que contiene en su primera posición la palabra y en la segunda el significado de ésta. Cada línea del archivo del diccionario tiene el siguiente formato:

palabra : categoria1 . [rol1 :] significado1 [+ rol2 : significado1 ...] [; categoria2 . [rol1 :] significado1 [+ rol2 : significado1 ...] ...]

```
lineaArchivo = self.archivo.get_linea()  
palabraSignificado = lineaArchivo.split(': ')  
palabraSignificado = palabraSignificado[0:1] +  
                    [': '.join(palabraSignificado[1:])] 
```

Para la palabra encontrada se crea un índice en el diccionario y con este índice se inicializa un subdiccionario.

```
self.diccionario[palabraSignificado[0]] = KeyInsensitiveDict()
```

A una segunda variable temporal se le asigna una lista en donde cada elemento contiene lo que está separado por ';' en el diccionario, es decir, cada una de las categorías. Así, `listacategoriasSeparadas` estará compuesta por elementos con el siguiente formato:

categoria . [rol1 :] significado1 [+ rol2 : significado1 ...]

```
listaCategoriasSeparadas = palabraSignificado[1].split('; ')
```

En el siguiente ciclo `categoriaSeparada` es cargada con cada uno de los valores de `listaCategoriasSeparadas`.

```
for categoriaSeparada in listaCategoriasSeparadas:
```

Una tercera variable temporal es creada, a esta se le asigna una lista que contiene en su primera posición la categoría y en la segunda los significados para esta categoría. El formato es el siguiente:

[rol1 :] significado1 [+ rol2 : significado1 ...]

Por categorías se entiende la clasificación sintáctica de la palabra, esto es, verbo, sustantivo, adjetivo, etc...

```
categoriaSignificado = categoriaSeparada.split('. ')
```

Para cada una de las categorías se crea un nuevo subdiccionario.

```
self.diccionario[palabraSignificado[0]]\  
[etiquetasBrown[categoriaSignificado[0]]] =\  
KeyInsensitiveDict()
```

Una cuarta variable temporal es cargada con una lista en donde cada elemento contiene lo que está separado por '+' en el diccionario, es decir, cada una de los roles y su correspondiente significado. Así, `listaRolesSeparados` estará compuesta por elementos con el siguiente formato:

[rol1 :] significado1

Por rol se entiende la clasificación semántica de la palabra, por ejemplo: *Mus --> Música, Zoo --> Zoología*

```
listaRolesSeparados = categoriaSignificado[1].split('+ ')
```

Variable que marca el primer rol encontrado.

```
primerRol = ''
```

En el siguiente ciclo `rolSeparado` es cargada con cada uno de los valores de `listaRolesSeparados`.

```
for rolSeparado in listaRolesSeparados:
```

Una última variable temporal es creada, a esta se le asigna una lista que contiene en su primera posición el género y en la segunda el texto correspondiente a los significados para este rol.

```
rolSignificado = rolSeparado.split(': ')
```

En este *if* se tiene en cuenta que es posible que algunos significados no tengan rol, esto ocurre cuando `rolSeparado` es igual a `rolSignificado`, cuando esto ocurre significa que no se encontró ningún símbolo ':', significando esto que la palabra no tiene información de roles.

```
if ([rolSeparado] == rolSignificado):
```

Se carga la variable `rol` con el valor *'NOROL'* significando que no tiene rol.

```
rol = 'NOROL'
```

La variable `indice` se carga con el valor de 0 indicando que los significados se encuentran en la posición 0 de `rolSignificado`.

```
indice = 0
```

Si encontró un arroba en `rolSignificado[0]` queda el rol.

```
else:  
    rol = rolSignificado[0]
```

La variable `indice` se carga con el valor de 1 indicando que los significados se encuentran en la posición 1 de `rolSignificado`.

```
indice = 1
```

Se carga en `primerRol` el valor del primer rol encontrado.

```
if primerRol == '':  
    primerRol = rol
```

Si ya existe un rol del mismo tipo se concatenan los significados, de lo contrario, se asignan directamente.

```

if self.diccionario[palabraSignificado[0]]\
[etiquetasBrown[categoriaSignificado[0]]].has_key(rol):
    self.diccionario[palabraSignificado[0]]\
    [etiquetasBrown[categoriaSignificado[0]]][rol]\
    = self.diccionario[palabraSignificado[0]]\
    [etiquetasBrown[categoriaSignificado[0]]][rol] +
    rolSignificado[indice].split(', ')[0]
else:
    self.diccionario[palabraSignificado[0]]\
    [etiquetasBrown[categoriaSignificado[0]]][rol]\
    = rolSignificado[indice].split(', ')[0]

```

Se agrega el rol por defecto 'NOROL' en caso de que no exista.

```

if not self.diccionario[palabraSignificado[0]]\
[etiquetasBrown[categoriaSignificado[0]]].\
has_key('NOROL'):
    self.diccionario[palabraSignificado[0]]\
    [etiquetasBrown[categoriaSignificado[0]]]['NOROL']\
    = self.diccionario[palabraSignificado[0]]\
    [etiquetasBrown[categoriaSignificado[0]]][primerRol]

```

Finalmente se retorna el diccionario.

```

return self.diccionario

```

Adicionalmente la clase tiene funciones para obtener información específica.

La función `tiene_palabra` examina si el diccionario tiene una palabra a evaluar, recibe como parámetro la palabra a buscar.

```

def tiene_palabra(self, palabra):
    if self.diccionario.has_key(palabra):
        return True
    else:
        return False

```

La función `tiene_categoria` examina si para una palabra dada del diccionario existe una categoría, recibe como parámetro la palabra y la categoría a buscar.

```

def tiene_categoria(self, palabra, categoria):
    if self.tiene_palabra(palabra):
        if self.diccionario[palabra].has_key(categoria):
            return True
        return False

```

La función `tiene_rol` examina si existe un rol para una categoría existente en una palabra, recibe como parámetro la palabra, categoría y rol a buscar.

```
def tiene_rol(self, palabra, categoria, rol):
    if self.tiene_categoria(palabra, categoria):
        if self.diccionario[palabra][categoria].has_key(rol):
            return True
        return False
```

La función `get_diccionario` retornar el contenido del diccionario.

```
def get_diccionario(self):
    return self.diccionario
```

La función `get_palabra` retorna las palabras del diccionario.

```
def get_palabras(self):
    return self.diccionario.keys()
```

La función `get_categorias` retornar las diferentes categorías de una palabra, recibe como parámetro la palabra a buscar.

```
def get_categorias(self, palabra):
    if self.tiene_palabra(palabra):
        return self.diccionario[palabra].keys()
    else:
        return []
```

La función `get_rol` retornar los diferentes roles para una categoría de una palabra, recibe como parámetro la palabra y la categoría a buscar.

```
def get_rol(self, palabra, categoria):
    if self.tiene_categoria(palabra, categoria):
        return self.diccionario[palabra][categoria].keys()
    return []
```

La función `get_rol` retornar el contenido de un rol, para una categoría dentro de una palabra, recibe como parámetro la palabra, categoría y rol de los cuales se desea obtener el significado.

```
def get_rol(self, palabra, categoria, rol):
    if self.tiene_rol(palabra, categoria, rol):
        return self.diccionario[palabra][categoria][rol]
    return ""
```

Como se pudo observar la carga del diccionario en el arreglo asociativo es un poco complicada y algorítmicamente compleja. Pero lo más importante es que la utilización del mismo es muy sencilla, aspecto muy importante para la interacción entre el diccionario y el usuario.

Este capítulo es de suma importancia en el proceso de traducción ya que suministra la mayor parte de la información para el proceso y se requiere su uso en las etapas posteriores, es decir, para realizar análisis morfológico, sintáctico, semántico y síntesis al lenguaje destino el diccionario es una pieza fundamental.

Habiendo definido el diccionario, el siguiente paso es realizar análisis morfológico, este tema se abordará en el capítulo siguiente. Este análisis consiste en aprovechar el diccionario para etiquetar gramaticalmente los tokens recibidos de la tokenización.

6. ANÁLISIS MORFOLÓGICO

6.1. INTRODUCCIÓN

En el capítulo anterior se abordó el tema de los diccionarios, que son de gran importancia ya que el análisis morfológico se apoya en dichos diccionarios, para etiquetar gramaticalmente cada uno de los tokens que recibe de la etapa de tokenización. Cabe anotar que los diccionarios no son parte del proceso de traducción, son un apoyo para el mismo.

El **análisis morfológico** es el proceso siguiente a la tokenización y permite identificar los lexemas y morfemas presentes en los tokens, es decir, cada palabra debe ser analizada para identificar las palabras derivativas. Los diccionarios usados en los sistemas de TA no tienen el significado de palabras derivadas de palabras más simples. Las palabras derivadas deben ser identificadas por el analizador. Las formas verbales y los plurales son las palabras derivadas más comunes. Además el diccionario le brinda al analizador morfológico la información gramatical adecuada para etiquetar los tokens y prepararlos para la siguiente etapa del proceso de traducción.

Dentro de la lingüística, la **morfología** estudia la estructura interior de las palabras o su configuración interna, y las clases de palabras a las que las diferentes estructuras internas dan lugar, así como la formación o construcción de nuevas palabras (neologismos). La palabra morfología (del griego *morphē*: forma) fue introducida en el siglo XIX.

A diferencia de la **sintaxis**, que toma las palabras como unidades para construir oraciones agrupándolas según su función en la estructura oracional, la morfología se interesa por la estructura interna de las palabras, e intenta descubrir las reglas que gobiernan la formación de palabras a partir de unidades menores.

El **monema** es la mínima unidad lingüística que posee significante y significado. El **significado** que es la idea o contenido que tenemos en la mente de cualquier palabra conocida, y el **significante** que es el conjunto de sonidos o letras con que se transmite el contenido de esa palabra conocida. Cuando una palabra no puede descomponerse en **unidades significativas** menores, se trata de un **monema**. Así pues, una palabra podrá estar constituida por uno o más monemas. Estos pueden ser de dos tipos principalmente, monemas léxicos o **lexemas** los cuales son los portadores del significado léxico y monemas morfemáticos o **morfemas** que son los portadores del significado gramatical.

6.2. ASPECTOS TEÓRICOS

6.2.1. Lexemas

Son los monemas que comúnmente se conocen como raíces y contienen esencialmente el significado de la palabra. Constituyen casi siempre la parte invariable de la misma y la de significado más concreto. Como aportan el significado fundamental de la palabra, se dice que son como la raíz. Forman la parte más numerosa del diccionario y su número en toda lengua es siempre muy superior al de los morfemas.

Cuando un lexema coincide con una palabra, como *nunca* o *cielo*, se trata de palabras primitivas. Si una palabra incluye varios lexemas, nos encontramos ante una palabra compuesta, como *limpiacristales* o *correveidile*.

6.2.2. Morfemas

Son los monemas que constituyen la parte variable de la palabra y cuyo significado es más abstracto y de carácter meramente lingüístico y gramatical. No son autónomos sino que se presentan siempre asociados a lexemas. Por ejemplo, en el vocablo *leonas*, los morfemas son *-a* (morfema que indica género femenino) y *-s* (morfema que indica número plural)

Existen distintos tipos: **libres**, **trabados** y el **morfema cero**.

6.2.2.1. Morfemas trabados o dependientes

Son aquellos que necesitan unirse a un lexema para tener significado. Entre ellos están: sufijos que son aquellos que se ponen después del lexema, prefijos que son los que se anteponen al lexema, obsérvese el ejemplo de sufijo y prefijo en la Figura 7. También se habla de interfijos que son los que van en el interior del lexema, por ejemplo en la lengua Bontoc de las Filipinas utiliza un infijo *um* para cambiar adjetivos y sustantivos en verbos. Por ejemplo la palabra *fikas*, que significa *fuerte* se transforma en el verbo *sean fuertes* adicionando el infijo *um* y la palabra quedaría *f-um-ikas*.

6.2.2.2. Morfemas libres o independientes

Son aquellos que no necesitan ir unidos a ningún lexema, sino que forman por sí solos una palabra. Son morfemas independientes los determinantes, las preposiciones y las conjunciones. Casi todos ellos son átonos, o sea que no recae

el acento prosódico¹¹⁸ y en realidad no son autónomos, sino que tienen que relacionarse con otras palabras. Ejemplos: de, las, y.

6.2.2.3. Morfema cero

En algunos casos, la ausencia de un **segmento fonémico** para expresar un significado, se interpreta como morfema cero (usualmente expresado morfo Ø). Así, tanto en *niño* como en *niños*, aparecen tres morfemas: *niñ-o-s*, con los significados de niño, masculino y plural, y *niñ-o-Ø* con los de niño, masculino y singular. El objetivo de utilizar esta unidad ficticia es conservar un paralelismo entre morfema y significado. En ocasiones, también se utiliza el morfo Ø como alomorfo del morfema de plural: *crisis-Ø*, frente a *casa-s* o *mantel-es*.

Por ejemplo, la palabra *internacionalmente* tiene tres partes, como se muestra a continuación en la Figura 7:

Figura 7 Clasificación de los morfemas



Se pueden diferenciar tres morfemas, cada uno posee significado por si mismo; *inter* significa en medio o entre, mientras que *mente* significa en forma de. El morfema *nacional* es libre por que puede aparecer por si solo, como una palabra en su propia esencia. Los morfemas de borde tienen que ser unidos a un morfema libre y no son palabras por si mismos.

6.2.3. Clasificación de los tipos de estructuras morfológicas¹¹⁹

El inglés, desde el punto de vista morfológico, es mas bien un lenguaje directo. La deducción se hace en cuanto que otros lenguajes se comportan en muchas ocasiones de diferentes formas y este es básicamente el esquema de clasificación

¹¹⁸ Forma de articulación de la voz al hablar mediante la cual se destaca una sílaba de una palabra respecto a las demás

¹¹⁹ Este apartado es una adaptación al español de: HANCOX, Peter P. H.. "MORPHOLOGICAL ANALYSIS: THE CLASSIFICATION OF MORPHOLOGICAL STRUCTURAL TYPES". 1 Jun 2005. <http://www.cs.bham.ac.uk/~pjh/sem1a5/pt2/pt2_intro_morphology.html>.

que se maneja. La lingüística desde sus inicios estuvo muy interesada en generar familias de lenguaje a manera de árbol para identificar la descendencia de los mismos, y sin embargo, nunca fue posible reconstruir lenguajes perdidos. La estructura morfológica es una buena forma de agrupar los lenguajes. Se pueden diferenciar tres clases principales.

6.2.3.1. Lenguajes aislados

Las palabras en un lenguaje aislado son invariables. Para decirlo de otra forma se componen de morfemas libres y no hay morfemas para indicar el número gramatical (ejemplo plural) o el tiempo (pasado, presente, futuro). El mandarín es un ejemplo de tal lenguaje.

6.2.3.2. Lenguajes aglutinantes

En estos lenguajes se busca combinar las palabras simples sin cambiar la forma para expresar ideas compuestas. Los ejemplos se basan generalmente en turco o swahili. Lo importante de estos lenguajes es notar cómo los morfemas representan una unidad del significado y cómo siguen siendo absolutamente identificables dentro de la estructura de las palabras. Totalmente lo contrario a lo que sucede con los lenguajes aislados.

6.2.3.3. Lenguajes que se descomponen

Las palabras en estos idiomas se presentan en diversas formas y es posible romper las palabras en unidades más pequeñas y etiquetarlas. Sin embargo, el resultado es confuso y contradictorio. Los ejemplos generalmente se basan en latín y se atribuyen al conocimiento gramatical latino del caso. Como ejemplo simple, el latín para la frase en inglés *I love* (en español Yo amo) es *amo*. Esto significa que la *o* final se utiliza para expresar los significados, primera persona, el presente del singular, y también otros significados.

Esta clasificación tiene solamente tres clases. Los cuestionamientos que se hacen giran en torno a la posibilidad de agrupar a todos los lenguajes en estas tres clases. Desde un punto de vista del problema, es imposible encajonar cualquiera de los idiomas en alguna de las clases, porque cada lengua es impura, es decir, si se mira con detenimiento, se encontrará descomposición principalmente en idiomas aglutinantes, descomposición en idiomas que aíslan, aglutinación en idiomas que se descomponen, etcétera.

6.2.4. Proceso de análisis morfológico¹²⁰

Hay un número de procesos morfológicos que son más importantes que otros para **PLN**. Los que se relacionan a continuación son solo algunos.

6.2.4.1. Inflexión

Este proceso es muy importante ya que acudiendo al diccionario identifica la raíz de la que provienen las palabras y les asigna información gramatical muy relevante para el proceso siguiente de análisis sintáctico.

La **inflexión** es el proceso de cambiar la forma de una palabra de modo que exprese información tal como los accidentes gramaticales pero la categoría sintáctica de la palabra queda sin cambios. Entre los accidentes gramaticales están el **número** que indica si un elemento se refiere a una o mas unidades según sea singular o plural respectivamente, la **persona** que se utiliza para referirse al individuo que habla (primera persona), a aquel a quien habla (segunda persona) o a aquel o aquello de que se habla (tercera persona), el **caso** que es la posibilidad de variación de la forma del sustantivo, adjetivo, pronombre, etcétera, según la función sintáctica que desempeñen en la oración, por ejemplo la palabra *gordo* se identifica como un adjetivo, pero en algún caso se puede utilizar como sustantivo si se refiere a la forma de llamar a alguna persona, el **género** que es la propiedad por la que tiene los sustantivos, adjetivos, artículos y pronombres se clasifican en masculinos y femeninos, el **tiempo** que es cada grupo de formas verbales que indican respectivamente que la acción se desarrolla en el momento en que se habla, o es anterior o posterior a éste, el **modo** que expresa el punto de vista de la persona que habla con respecto con relación a la acción del verbo, entre otros.

Por ejemplo, la forma plural del sustantivo en inglés es formada generalmente de la forma singular agregando una *s*.

car / cars
table / tables
dog / dogs

En cada uno de estos casos, se mantiene la categoría sintáctica de la palabra sin cambiar. Es claro que existen excepciones a la regla mencionada anteriormente. Hay algunos grupos de sustantivos que forman plurales de diversas formas:

¹²⁰ Este apartado es una adaptación al español de: HANCOX, Peter P. H. "MORPHOLOGICAL ANALYSIS: Morphological processes". 1 Jun 2005. <http://www.cs.bham.ac.uk/~pjh/sem1a5/pt2/pt2_intro_morphology.html>.

wolf / wolves
knife / knives
switch / switches.

Si se piensa un poco mas se pueden encontrar formas plurales al parecer totalmente irregulares, por ejemplo:

foot / feet
child / children.

Los verbos ingleses son relativamente simples (comparado especialmente con idiomas como el finlandés que tiene cerca de 12.000 inflexiones verbales).

mow - raíz
mows - tercera persona singular, presente
mowed - pasado y participio pasado
mowing - gerundio

- Aspectos PLN

Los idiomas como francés o alemán tienen mucha más inflexión que el inglés y es así como se acostumbra incluir analizadores morfológicos en los sistemas que procesan estos idiomas. Los sistemas de PLN para el inglés no incluyen a menudo ningún proceso morfológico, especialmente si son sistemas en reducida escala. Los sistemas basados en inglés incluyen el análisis de la inflexión, las formas regulares de palabras se analizan usando una de las técnicas estándares (por ejemplo, autómatas finitos del estado), mientras que las excepciones (las palabras irregulares) son cada una enumerada individualmente. Esto significa que las formas regulares tienen que ser incorporadas al diccionario solamente una vez, lo que ahorra mucho espacio en la entrada de datos para que así el diccionario lleve mucha de la información sintáctica y semántica.

6.2.4.2. Derivación

La inflexión no cambia la categoría sintáctica de una palabra. La **derivación** cambia la categoría. Los lingüistas clasifican la derivación en inglés según induzca o no un cambio de la pronunciación. Por ejemplo, agregando el sufijo *ity* cambia la pronunciación de la palabra raíz *active* en la acentuación de la segunda sílaba, la palabra quedaría así *activity*. La adición del sufijo *al* a *approve* no cambia la pronunciación de la raíz, la palabra quedaría así: *approval*.

- Aspectos PLN

El objetivo de la morfología derivativa en sistemas de PLN es reducir el número de formas de palabras que se almacenarán. Así pues, si hay ya una entrada para la forma base del verbo *sing*, puede ser posible agregar reglas para construir los

sustantivos *singer* y *singers* sobre la misma entrada. El problema es que la detección de la derivación de *singer* de *sing* debe permitir también que el analizador morfológico pueda incluir información que es especial para *singer*. Esto parece un poco complicado, pero un ejemplo lo hará más claro. La adición de *er* a una palabra indica que es una persona que está emprendiendo la acción. Esta información semántica se debe agregar a la información almacenada de la palabra para la forma de la raíz *sing* para poder encontrar el significado correcto de la oración. Esto parece muy sencillo, pero supóngase las dos palabras siguientes: *recorder* y *dragster*. El uso del morfema *er* no necesariamente indica que alguien emprende la acción representada por la forma de la raíz.

En algún proceso lingüístico es probable deshacer la ambigüedad potencial, particularmente donde los seres humanos no contaban con ella. Los analizadores morfológicos derivativos pueden hacer esto absolutamente fácil, porque están procurando siempre reducir palabras a unidades más pequeñas. La palabra *really* se puede analizar alternamente con *really* y *re + ally*. Esta ambigüedad introducida se puede eliminar más adelante, en el proceso sintáctico pero no obstante, significa que tiene que haber más proceso (y por tanto un sistema más lento).

La morfología derivativa es particularmente útil para la traducción automática. La TA acertada tiene que procesar cantidades grandes de texto que pueden contener muchas palabras previamente no vistas. Algunas palabras son **neologismos** (es decir, palabras nuevas en una lengua). Si el analizador puede reducir estas palabras a su forma base, puede poder traducir eso y, en efecto, acuñar una nueva palabra en la lengua objetivo simplemente por reglas. Para dar un par de ejemplos: los neologismos tienen a menudo un nombre propio como su raíz. Si se sabe que Thatcherite y Majorism fueron formados de nombres propios podría permitir a un sistema de la TA traducirlos a un equivalente idiomático en la lengua objetivo.

6.2.4.3. Semi-afijos y formas combinadas

Los **semi-afijos** son los morfemas que están limitados pero que conservan palabra como garantía. Los ejemplos como: *anti-*, *counter-*, *-like* y *-worthy*. Se puede tener entonces combinaciones como:

anti-clockwise o anticlockwise
counter-example o counterexample
bird-like o birdlike
note-worthy o noteworthy

Combinando las formas existen más cuasi-palabras que semi-afijos y ocurren con frecuencia en literatura técnica, por ejemplo *Indo-European* o *gastro-enteritis*.

Algunas palabras se pueden componer enteramente de formas encuadradas, pero sin un morfema libre, ejemplo: *franco-phile*.

- Aspectos PLN

Como con morfología derivativa, los semi-afijos y formas combinadas se pueden analizar en sus morfemas y, como la morfología derivativa, puede ser utilizada para analizar palabras previamente no vistos o que no existían. La inscripción con guión es un problema particular para los idiomas como el inglés y el alemán y una comprensión de semi-afijo y formas combinadas puede contribuir a identificar puntos opcionales y probables de la inscripción con guión al procesar el texto.

6.2.4.4. Manejo de contracciones

Se puede optar por el manejo de las contracciones en el proceso de análisis sintáctico, ya en ese punto se tiene información suficiente para eliminar las ambigüedades que se presentan con las mismas.

Una contracción es un elemento que se comporta como un afijo y una palabra. Sin embargo, son absolutamente complicados en que son también parte de la formación de la palabra. A diferencia de otros fenómenos morfológicos, la contracción ocurre en una estructura sintáctica y su adjunción a las palabras no es parte de las reglas de la formación de la palabra como el resto de morfología.

El inglés tiene una contracción obvia, el 's usado para denotar el posesivo (conocido a veces como el caso gramatical del genitivo). Los lingüistas lo llaman un enclítico que significa que es una contracción que se adjunta a la derecha de la palabra, al igual que un sufijo.

La contracción 's se une a un independiente constitutivo específico de donde ocurre en la oración. En los dos ejemplos siguientes, 's se une primero a un sustantivo y después a una preposición:

the girl's penguin
the car I bumped into's headlight

La preocupación de los lingüistas es que el 's está producido del léxico para expresar la relación posesiva, pero su posición es determinada por la estructura sintáctica de la elocución.

El inglés también proporciona diversas formas de hacer contracciones. Algunas palabras se pueden reducir a una forma más corta. Por ejemplo: I am in Birmingham puede ser reducido a I'm in Birmingham y It is a spring chicken puede ser reducido a It's a spring chicken. Obsérvese que la palabra que es reducida tiene su propia categoría

sintáctica y la seguiría teniendo por si misma en cualquier análisis sintáctico de una oración.

- Aspectos PLN

Las contracciones son un problema interesante para PLN. Los sistemas PLN convencionales son modulares y tienen módulos de procesamiento morfológico, sintáctico y semántico distintos. Sin embargo, las contracciones como 's no pueden ser analizadas satisfactoriamente. Un analizador morfológico tiene que poder separar la contracción de su morfema unido, pero no puede hacer esto correctamente a menos que conozca la estructura sintáctica de la elocución. En una arquitectura convencional del sistema de PLN, la estructura sintáctica no está disponible para un analizador morfológico.

Hay varios métodos que se podrían utilizar para solucionar este problema, tal como pasar tantas alternativas como sea posible del analizador morfológico, o esperar que el análisis sintáctico resuelva estas ambigüedades. Otro método puede ser realizar análisis morfológico y sintáctico paralelos o quizás la morfología y la sintaxis son maneras imperfectas de describir la lengua y se debe encontrar un modelo descriptivo mejor.

6.2.5. Otras consideraciones

Cada palabra debe ser analizada para identificar las palabras derivativas. Los diccionarios usados en los sistemas de TA no tienen el significado de palabras derivadas de palabras más simples. Las palabras derivadas deben ser identificadas por el analizador. Las formas verbales y los plurales son las palabras derivadas más comunes.

6.2.5.1. La necesidad de realizar análisis morfológico

El **análisis morfológico** es un proceso muy importante ya que proporciona información acerca de los tokens con etiquetamiento, aspectos fundamentales en el proceso de traducción, ya que facilita la realización de análisis sintáctico y semántico.

El análisis morfológico es una técnica utilizada en el procesamiento de lenguaje natural. Cuando se piensa en análisis morfológico, se puede confundir con análisis sintáctico. Pero antes de poder realizar análisis sintáctico a un texto se debe contar con información especial sobre cada palabra en el texto. Por ejemplo, para analizar la oración *el gato persiguió la rata*, se debe saber que *el gato* es un sustantivo singular, *perseguir* es un verbo en tiempo pasado, etcétera. En inglés, tal información se puede obtener con un léxico que enumere todas las palabras

con su parte del habla como clasificar si es un sustantivo, un verbo, un adjetivo, entre otros e información **inflexional** como número y tiempo. El inglés tiene un sistema inflexional relativamente simple. Obsérvese que los sustantivos que se pueden contar tales como *cat* tienen solamente dos formas derivadas, singular y plural, y los verbos regulares tales como *persecute* tienen solamente cuatro formas derivadas: la forma base, la forma de *s*, la forma del *ed*, y la forma del *ing*. Pero un listado léxico exhaustivo no se puede utilizar tan fácilmente para otros idiomas, tales como finlandés, turco, y quechua, que pueden tener centenares de formas derivadas para cada sustantivo o verbo. Para tales idiomas, se debe construir un analizador morfológico que utilice el sistema morfológico de la lengua propiamente para computar la parte del habla y las categorías inflexionales de cualquier palabra.

Incluso para el inglés podría ser necesario. Aunque el inglés tiene un sistema inflexional limitado, tiene morfología derivativa muy compleja y productiva. Por ejemplo, de la raíz *compute* vienen las formas derivadas tales como *computer*, *computerize*, *computerization*, *recomputerize*, *noncomputerized*, etcétera. Es imposible enumerar exhaustivamente en un léxico todas las formas derivadas (incluyendo términos acuñados y palabras propias de cada lenguaje) que pudieron ocurrir en texto natural.

6.3. MODELAMIENTO DEL ANALIZADOR MORFOLÓGICO

Ya se explicó anteriormente que para un idioma determinado existen palabras raíces o lexemas y palabras derivadas de estos haciendo uso de reglas de derivación o inflexión. De tal forma, se puede definir lo siguiente:

Si una palabra w' puede ser originada a partir de un lexema w haciendo uso de una o varias reglas de derivación o inflexión morfológica R_i . Se dice entonces que w' se obtiene mediante composición morfológicas a partir de w .

$w' = R_i(w)$ Donde w es la palabra raíz y w' es la palabra obtenida a partir de w haciendo uso de la regla R_i . Por ejemplo:

Sea $w = cat$ y R_i la regla de formación de plural. Así pues w' será el plural de *cat* dando como resultado que $w' = cats$.

De igual manera se puede presentar casos de derivación compuesta. Es decir que una palabra se derive de una palabra que ya ha sido derivada de otra. Un ejemplo claro de derivación compuesta se presenta en la palabra *leonas*, ya que:

leonas = Plural(Femenino(Leon))

Ahora bien, en el proceso de análisis morfológico es necesario realizar el proceso inverso. Dada una palabra se requiere saber de que palabra se deriva esta. Por ejemplo si se tiene la palabra *leonas* debe poderse decir que corresponde a *leon* en *plural* y *femenino*. En el caso de *cats* debe tenerse como resultado *cat* en *plural*.

De tal manera que si se tiene una palabra w' y si aplicando a esta una función R_i^{-1} se obtiene una palabra raíz w (la cual debería estar en el diccionario), se dice entonces que posiblemente¹²¹ w' es originada a partir de un proceso de inflexión o derivación de w . Por ejemplo la palabra *spies*, puede ser un verbo conjugado en tercera persona en presente simple o un sustantivo en plural.

Como se acaba de mostrar dicho proceso puede presentar problemas de ambigüedad. Ya que una palabra derivada puede derivarse de diferentes raíces haciendo uso de diferentes reglas de composición morfológica, tal como en el ejemplo anterior para el verbo *save* la conjugación de presente simple en tercera persona da como resultado la misma palabra al aplicar el plural al sustantivo *save*.

Aún teniendo dicho problema de ambigüedad, todas las raíces w posibles así como las reglas R_i aplicadas para llegar a la palabra derivada w' deben ser desplegadas, ya que más adelante el proceso de análisis sintáctico se hará la desambiguación.

Es necesario tener en cuenta que si al aplicar una regla inversa R_i^{-1} a una de las palabras w' (las cuales en este caso son los tokens obtenidos en el proceso de tokenización) se obtiene una raíz w . Es importante verificar que al aplicar la regla R_i se obtenga nuevamente la palabra w' ya que sino es así podría afirmarse que w' se deriva de w a partir de la regla R_i siendo esto incorrecto.

Aún teniendo en cuenta lo anterior es posible que para idiomas particulares se presentes diferentes tipos de irregularidades. Por ejemplo el verbo *to be* en inglés no puede obtenerse fácilmente a partir de reglas de conjugación. De igual manera el verbo *ir* en español. Este tipo de irregularidades deben ser tenidas en cuenta en el momento de definir las reglas de composición morfológica.

¹²¹ Se dice posiblemente ya que en este paso no se tiene certeza que tal palabra realmente corresponda a dicha raíz. Esta información sólo se puede saber más adelante el proceso de análisis sintáctico.

Finalmente se recomienda el uso de etiquetas estandarizadas para las diferentes categorías de una palabra. Para esto se aconseja usar el *Brown Corpus Tag Set* (Véase el Anexo F). Por ejemplo para la palabra *bleats* debería ser identificada como un verbo conjugado en presente en tercera persona y que dicha palabra debe ser buscada en el diccionario como *bleat* y no como *bleats* ya que esta última es una palabra derivada que no se encuentra en el diccionario.

6.4. IMPLEMENTACIÓN DEL ANALIZADOR MORFOLÓGICO

El siguiente es el contenido del archivo `Morfologico.py`, el cual es el código fuente del analizador morfológico realizado.

```
from traductor.librerias.DiccionarioInsensitive import KeyInsensitiveDict
from traductor.tokenizacion import Tokenizador
from traductor.diccionario import Diccionario
import re
import string

class AnalizadorMorfologico:
    def __init__(self, objTokenizador, objDiccionario):
        self.tokenizador = objTokenizador
        self.diccionario = objDiccionario
        self.diccionarioMorfologico = KeyInsensitiveDict()

    def analizar(self):
        self.diccionarioMorfologico = KeyInsensitiveDict()
        for token in self.tokenizador.get_tokens():
            etiquetasMorfologico = self._etiquetar_palabra(token)
            self.diccionarioMorfologico[token] = etiquetasMorfologico
        return self.diccionarioMorfologico

    def _etiquetar_palabra(self, palabra):
        plurales = ['ies', 'es', 's']
        pronombres3Persona = KeyInsensitiveDict({'he': '', 'she': '',
                                                  'it': '', 'thee': ''})
        irregularidadesToBe = KeyInsensitiveDict({'am': 'BEM', 'are': 'BER',
                                                  'is': 'BEZ', 'being': 'BEG'})

        vocales = 'aeiou'
        consonantes = 'bcdfghjklmnpqrstvxyz'
        palabraReal = palabra
        etiquetasMorfologico = KeyInsensitiveDict()
        if self.diccionario.tiene_palabra(palabraReal):
            for etiquetaBrown in self.diccionario.get_categorias(palabraReal):
                if etiquetaBrown == 'PPS' and not pronombres3Persona.has_key(palabraReal):
                    etiquetasMorfologico[etiquetaBrown+'S'] = (palabraReal,
                                                                etiquetaBrown)
            else:
```

```

        etiquetasMorfologico[etiquetaBrown] = (palabraReal,
                                                etiquetaBrown)

patron = re.compile('s$', re.I)
if (len(patron.findall(palabra)) > 0):
    for fin in plurales:
        palabraReal = palabra[:-len(fin)]
        if fin == 'ies':
            palabraReal = palabraReal + 'y'
            if self.diccionario.tiene_categoria(palabraReal, 'VB'):
                etiquetasMorfologico['VBZ'] = (palabraReal, 'VB')
            if self.diccionario.tiene_categoria(palabraReal, 'NN'):
                etiquetasMorfologico['NNS'] = (palabraReal, 'NN')
else:
    patron = re.compile('ing$', re.I)
    if (len(patron.findall(palabra)) > 0):
        palabraReal = palabra[:-3]
        patron = re.compile('[' + consonantes + ']{1}[' + vocales +
                            ']{1}[' + consonantes + ']{2}$', re.I)
        if palabraReal[-2]==palabraReal[-1] and
            (len(patron.findall(palabraReal)) > 0):
            palabraReal = palabraReal[:-1]
        else:
            patron = re.compile('ck$', re.I)
            if (len(patron.findall(palabraReal)) > 0):
                palabraReal = palabraReal[:-1]
            else:
                patron = re.compile('i$', re.I)
                if (len(patron.findall(palabraReal)) > 0):
                    palabraReal = palabraReal[:-1] + 'y'
                else:
                    if not self.diccionario.tiene_palabra(palabraReal):
                        palabraReal = palabraReal + 'e'
                    if self.diccionario.tiene_categoria(palabraReal, 'VB'):
                        etiquetasMorfologico['VBG'] = (palabraReal, 'VB')
if (etiquetasMorfologico.keys() == [] or
    irregularidadesToBe.has_key(palabra)):
    if (irregularidadesToBe.has_key(palabra)):
        etiquetasMorfologico[irregularidadesToBe[palabra]] = ('be',
                                                                'VB')
    else:
        patronPuntuacion1 = re.compile('((\.{3})|[\.\.:;\?!])')
        patronPuntuacion2 = re.compile('([-\\[\\]\\(\\),])')
        if patronPuntuacion1.search(palabraReal):
            etiquetasMorfologico['.'] = (palabraReal, '.')
        else patronPuntuacion2.search(palabraReal):
            etiquetasMorfologico[palabraReal] = (palabraReal,
                                                  palabraReal)
        else:
            etiquetasMorfologico['NP'] = (palabraReal, 'NP')
return etiquetasMorfologico

def tiene_palabra(self, palabra):

```

```

        if self.diccionarioMorfologico.has_key(palabra):
            return True
        else:
            return False

    def tiene_categoria(self, palabra, categoria):
        if self.tiene_palabra(palabra):
            if self.diccionarioMorfologico[palabra].has_key(categoria):
                return True
            else:
                return False

    def get_diccionario(self):
        return self.diccionarioMorfologico

    def get_palabras(self):
        return self.diccionarioMorfologico.keys()

    def get_categorias(self, palabra):
        if self.tiene_palabra(palabra):
            return self.diccionarioMorfologico[palabra].keys()
        else:
            return []

    def get_palabra_real(self, palabra, categoria):
        if self.tiene_categoria(palabra, categoria):
            return self.diccionarioMorfologico[palabra][categoria][0]
        return ""

    def get_categoria_real(self, palabra, categoria):
        if self.tiene_categoria(palabra, categoria):
            return self.diccionarioMorfologico[palabra][categoria][1]
        return ""

```

A continuación se ofrece una explicación de la manera en que el programa citado anteriormente realiza el análisis morfológico.

Se importa la librería `string` para el manejo de cadenas de caracteres y la librería `re` para el manejo de las expresiones regulares. De igual forma se importa la librería `DiccionarioInsensitive` para poder manejar las llaves independientemente de si están en mayúsculas o minúsculas. Además se importa la clase `Tokenizador` y la clase `Diccionario`.

```

from traductor.librerias.DiccionarioInsensitive import KeyInsensitiveDict
from traductor.tokenizacion import Tokenizador
from traductor.diccionario import Diccionario
import re
import string

```

Se define la case `AnalizadorMorfologico` y su constructor, así mismo el constructor inicializa un objeto de tipo `Tokenizador` y otro objeto de tipo `Diccionario`, además se inicializa la variable `diccionarioMorfologico`.

```
class AnalizadorMorfologico:
    def __init__(self, objTokenizador, objDiccionario):
        self.tokenizador = objTokenizador
        self.diccionario = objDiccionario
        self.diccionarioMorfologico = KeyInsensitiveDict()
```

Se define la función principal `analizar` que se encargara de realizar el análisis morfológico. Retomando el apartado 4.2.1.2 en donde se estableció la diferencia entre un token y un **tipo**, se retorna la lista de tipos pues el arreglo asociativo no permite ocurrencias múltiples de las llaves, y esto precisamente es la lista de tipos, la cual omite las ocurrencias múltiples de la lista de tokens, además no es necesario tener información repetida del etiquetamiento de las palabras.

Para cada uno de los tokens recorre la función `etiquetar_palabra`, etiquetando cada palabra según su morfología. Es necesario tener en cuenta que el arreglo asociativo retornado no contiene el orden de los tokens. De igual manera si hay tokens repetidos sólo tiene en cuenta uno. Esto no representa ningún problema ya que para dos tokens iguales las etiquetas serán las mismas, posteriormente se debe hacer un análisis (sintáctico) para determinar cual de las etiquetas es válida. Al final se retorna un arreglo asociativo `diccionarioMorfologico` en donde las llaves son los tipos de la lista de tokens recibidos y el valor asociado a cada llave es la lista de etiquetas devueltas por la función `etiquetar_palabra`.

```
def analizar(self):
    self.diccionarioMorfologico = KeyInsensitiveDict()
    for token in self.tokenizador.get_tokens():
        etiquetasMorfologico = self._etiquetar_palabra(token)
        self.diccionarioMorfologico[token] = etiquetasMorfologico
    return self.diccionarioMorfologico
```

Para la implementación de un prototipo que pueda mostrar la funcionalidad de la teoría esbozada anteriormente se realizó una función desarrollada en *Python* `etiquetar_palabra`, la cual recibe como parámetro la palabra a la cual se le realizará el proceso de análisis. Esta función descompone la palabra en los morfemas que contiene y retorna una lista `etiquetas` que contiene las diferentes etiquetas que son morfológicamente válidas.

```
def _etiquetar_palabra(self, palabra):
```

En la primera línea se define una lista `plurales`, estos son los diferentes plurales que se pueden presentar, se definen de esta manera porque su tratamiento es similar salvo por una irregularidad que se presenta con palabras terminadas en *y*. Con esta lista y ejecutando un ciclo se realiza el tratamiento para cada uno de ellos. Así mismo se definen los pronombres en tercera persona `pronombres3Persona` y las irregularidades del verbo *to be* `irregularidadesToBe`, ya que tienen tratamiento especial.

```
plurales = ['ies', 'es', 's']
pronombres3Persona = KeyInsensitiveDict({'he': '', 'she': '',
                                         'it': '', 'thee': ''})
irregularidadesToBe = KeyInsensitiveDict({'am': 'BEM', 'are': 'BER',
                                         'is': 'BEZ', 'being': 'BEG'})
```

En las líneas siguientes se realiza una definición de las listas de vocales y consonantes, para ser utilizadas posteriormente en las reglas gramaticales.

```
vocales = 'aeiou'
consonantes = 'bcdfghjklmnpqrstvxyz'
```

Se crea una variable llamada `palabraReal`, la cual en un principio tiene el valor de la palabra a ser etiquetada, esta variable será modificada con el fin de buscar coincidencias de ésta en el diccionario identificando las etiquetas que corresponden a la palabra original, cuyo valor es recibido como parámetro en la variable `palabra`. Se hace necesario trabajar sobre otra variable ya que es posible que una misma palabra tenga diferentes orígenes morfológicos y por lo tanto diferentes etiquetas. También es creada la lista de etiquetas `etiquetasMorfológico`, la cual en un principio se crea vacía.

```
palabraReal = palabra
etiquetasMorfológico = KeyInsensitiveDict()
```

La siguiente condición examina si la palabra tal como viene se encuentra en el diccionario, de ser así se adiciona a la lista de `etiquetasMorfológico`, la etiqueta *brown* que corresponda a la ocurrencia en el diccionario y la palabra real, es decir, se conforma una tupla (`palabraReal`, `etiquetaBrown`). Con la condición se evalúan los pronombres que no pertenecen a la tercera persona para agregar una *s* al final de la etiqueta y así conservar el formato de las etiquetas *brown*.

```
if self.diccionario.tiene_palabra(palabraReal):
    for etiquetaBrown in
        self.diccionario.get_categorias(palabraReal):
```

```

if etiquetaBrown == 'PPS' and not
pronombres3Persona.has_key(palabraReal):
    etiquetasMorfologico[etiquetaBrown+'S'] = (palabraReal,
                                                etiquetaBrown)
else:
    etiquetasMorfologico[etiquetaBrown] = (palabraReal,
                                            etiquetaBrown)

```

Con la condición siguiente se realiza el tratamiento a los plurales. Si la palabra termina con 's' es candidata a ser un sustantivo o adjetivo en plural o un verbo conjugado en tercera persona. Mediante un ciclo *for* se carga en la variable *fin* cada uno de las terminaciones de los plurales para realizar el análisis. Posteriormente en la variable *palabraReal* quedará la palabra original sin la terminación dada por el valor de la variable *fin*. Es decir que se ha eliminado la terminación que denotaba plural. Luego se inicia una condición, la cual maneja una irregularidad, en caso de que la terminación de plural fuese *ies* es necesario agregar una *y* para cumplir con la regla gramatical que advierte que una palabra terminada en *y* debe ser reemplazada por *ies*. Ejemplo: *spy*, plural *spies*. Si después de los cambios realizados *palabraReal* se encuentra en el diccionario se ha encontrado una palabra derivada. En el caso de sustantivos y adjetivos es una palabra en plural, si por el contrario es un verbo, éste estaba conjugado para la tercera persona en presente.

```

patron = re.compile('s$', re.I)
if (len(patron.findall(palabra)) > 0):
    for fin in plurales:
        palabraReal = palabra[:-len(fin)]
        if fin == 'ies':
            palabraReal = palabraReal + 'y'
            if self.diccionario.tiene_categoria(palabraReal, 'VB'):
                etiquetasMorfologico['VBZ'] = (palabraReal, 'VB')
            if self.diccionario.tiene_categoria(palabraReal, 'NN'):
                etiquetasMorfologico['NNS'] = (palabraReal, 'NN')

```

Se emplea otra condición la cual evalúa si la palabra termina con *ing*, en caso de que se cumpla es posible que se trate de un verbo + *ing*. De ser así Se suprime la terminación *ing* para realizar el análisis. Si la palabra tiene las dos últimas letras repetidas y además termina de la forma *CVCC* donde *C* es cualquier consonante y *V* cualquier vocal, entonces se suprime la última consonante. Para cumplir la regla gramatical. Ejemplo: *snag*, gerundio *snagging*. Si lo anterior no ocurrió y se cumple que la palabra termina en *ck* se suprime la *k*. Ejemplo: *frollic*, gerundio *frollicking*. Ahora bien, si lo anterior no ocurrió y se cumple que la palabra termina en *i* es posible que esta haya sido un reemplazo de una *y*. Posteriormente, si a pesar de los cambios realizados la palabra no se encuentra en el diccionario se agrega la letra *e* que es suprimida al final en algunas ocasiones al agregar *ing*. Finalmente, si

se encuentra un verbo que concuerde con la palabra, se ha encontrado un verbo + *ing*. Al ocurrir esto se agrega a la lista de etiquetas la etiqueta *VBG*.

```

else:
    patron = re.compile('ing$', re.I)
    if (len(patron.findall(palabra)) > 0):
        palabraReal = palabra[:-3]
        patron = re.compile('[' + consonantes + ']{1}[' + vocales +
                             ']{1}[' + consonantes + ']{2}$', re.I)
        if palabraReal[-2]==palabraReal[-1] and
            (len(patron.findall(palabraReal)) > 0):
            palabraReal = palabraReal[:-1]
        else:
            patron = re.compile('ck$', re.I)
            if (len(patron.findall(palabraReal)) > 0):
                palabraReal = palabraReal[:-1]
            else:
                patron = re.compile('i$', re.I)
                if (len(patron.findall(palabraReal)) > 0):
                    palabraReal = palabraReal[:-1] + 'y'
                else:
                    if not self.diccionario.tiene_palabra(palabraReal):
                        palabraReal = palabraReal + 'e'
            if self.diccionario.tiene_categoria(palabraReal, 'VB'):
                etiquetasMorfologico['VBG'] = (palabraReal, 'VB')

```

Si `etiquetasMorfologico` no tiene contenido o la palabra pertenece al verbo *To Be* se ingresa al siguiente cuerpo de código.

```

if (etiquetasMorfologico.keys() == [] or
    irregularidadesToBe.has_key(palabra)):

```

Si la palabra hace parte de las palabras del verbo *To Be* se realiza el manejo de las irregularidades presentadas con el verbo *To Be*, se agrega a `etiquetasMorfo`.

```

if (irregularidadesToBe.has_key(palabra)):
    etiquetasMorfologico[irregularidadesToBe[palabra]] = ('be',
                                                            'VB')

```

Si esto no se cumplió quiere decir que la palabra no hace parte del diccionario, en este punto la palabra sólo puede ser un signo de puntuación o una palabra desconocida, para este último caso la palabra es catalogada como un *NP* (Nombre propio).

Las etiquetas *Brown* establecen diferencia entre los signos de puntuación para terminar frases como: puntos suspensivos '...', punto '.', dos puntos ':', punto y coma ';', interrogación '?' y admiración '!', los cuales son etiquetados con punto

‘.’ y los otros signos de puntuación los cuales su etiqueta corresponde al mismo signo como: guión ‘-’, paréntesis ‘()’, corchetes ‘[]’ y la coma ‘,’ Es por esto que se construyen dos expresiones regulares para identificar cada uno de estos.

```
else:
    patronPuntuacion1 = re.compile('((\.{3})|[\.:;\?!])')
    patronPuntuacion2 = re.compile('([-\[\\\]\(\),])')
    if patronPuntuacion1.search(palabraReal):
        etiquetasMorfologico['.'] = (palabraReal, '.')
    else patronPuntuacion2.search(palabraReal):
        etiquetasMorfologico[palabraReal] = (palabraReal,
                                              palabraReal)
else:
    etiquetasMorfologico['NP'] = (palabraReal, 'NP')
```

Posteriormente se retorna la lista de etiquetas.

```
return etiquetasMorfologico
```

La clase `AnalizadorMorfologico` tiene además algunas funciones para fines específicos.

La función `tiene_palabra` que examina si el diccionario tiene una palabra a evaluar, recibe como parámetro la palabra a buscar.

```
def tiene_palabra(self, palabra):
    if self.diccionarioMorfologico.has_key(palabra):
        return True
    else:
        return False
```

La función `tiene_categoria` que examina si para una palabra dada del diccionario existe una categoría, recibe como parámetro la palabra y categoría a buscar.

```
def tiene_categoria(self, palabra, categoria):
    if self.tiene_palabra(palabra):
        if self.diccionarioMorfologico[palabra].has_key(categoria):
            return True
        else:
            return False
```

La función `get_diccionario` que retornar el resultado del análisis morfológico.

```
def get_diccionario(self):
```

```
return self.diccionarioMorfologico
```

La función `get_palabras` que retornar las palabras del diccionario.

```
def get_palabras(self):  
    return self.diccionarioMorfologico.keys()
```

La función `get_categorias` que retornar las diferentes categorías de una palabra, recibe como parámetro la palabra a la que se le desea conocer sus categorías.

```
def get_categorias(self, palabra):  
    if self.tiene_palabra(palabra):  
        return self.diccionarioMorfologico[palabra].keys()  
    else:  
        return []
```

La función `get_palabra_real` que retornar la palabra real a buscar en el diccionario para una palabra y categoría, recibe como parámetro la palabra y categoría a buscar.

```
def get_palabra_real(self, palabra, categoria):  
    if self.tiene_categoria(palabra, categoria):  
        return self.diccionarioMorfologico[palabra][categoria][0]  
    return ""
```

La función `get_categoria_real` que retornar la categoría real a buscar en el diccionario para una palabra y categoría, recibe como parámetro la palabra y categoría a buscar.

```
def get_categoria_real(self, palabra, categoria):  
    if self.tiene_categoria(palabra, categoria):  
        return self.diccionarioMorfologico[palabra][categoria][1]  
    return ""
```

A continuación se muestra la funcionalidad del prototipo descrito anteriormente, para la siguiente entrada:

```
words = ['I', 'she', 'you', 'cats', 'spies', 'fishes', 'drys', 'happy', 'save', 'working', 'snagging',  
'frolicking', 'moving', 'moves', 'crying', 'visaing', 'Sebastian', '...', '?', '.', ',', '(', 'and', 'my', 'is',  
'are', 'am']
```

Se obtiene un arreglo asociativo del siguiente tipo:

```
{'token1': {'etiqueta1': (palabraReal, etiquetaReal), 'etiqueta2': (palabraReal, etiquetaReal)...},
'token2': {'etiqueta1': (palabraReal, etiquetaReal), 'etiqueta2': (palabraReal, etiquetaReal), ...},
...}
```

Para visualizarlo mucho mejor se ha optado por presentarlo en una manera mucho más cómoda para el lector:

```
Token: 'and' Etiqueta: 'CC' (Palabra Real, etiqueta Real): ('and', 'CC')
Token: 'is' Etiqueta: 'BEZ' (Palabra Real, etiqueta Real): ('be', 'VB')
Token: 'am' Etiqueta: 'BEM' (Palabra Real, etiqueta Real): ('be', 'VB')
Token: 'are' Etiqueta: 'BER' (Palabra Real, etiqueta Real): ('be', 'VB')
Token: 'moves' Etiqueta: 'VBZ' (Palabra Real, etiqueta Real): ('move', 'VB')
      'moves' Etiqueta: 'NNS' (Palabra Real, etiqueta Real): ('move', 'NN')
Token: 'drys' Etiqueta: 'VBZ' (Palabra Real, etiqueta Real): ('dry', 'VB')
Token: 'working' Etiqueta: 'VBG' (Palabra Real, etiqueta Real): ('work', 'VB')
Token: '(' Etiqueta: '(' (Palabra Real, etiqueta Real): ('(', '(')
Token: ',' Etiqueta: ',' (Palabra Real, etiqueta Real): (',', ',')
Token: '.' Etiqueta: '.' (Palabra Real, etiqueta Real): ('.', '.')
Token: 'cats' Etiqueta: 'NNS' (Palabra Real, etiqueta Real): ('cat', 'NN')
      'crying' Etiqueta: 'VBG' (Palabra Real, etiqueta Real): ('cry', 'VB')
      'crying' Etiqueta: 'NN' (Palabra Real, etiqueta Real): ('crying', 'NN')
      'crying' Etiqueta: 'JJ' (Palabra Real, etiqueta Real): ('crying', 'JJ')
Token: 'you' Etiqueta: 'PPSS' (Palabra Real, etiqueta Real): ('you', 'PPS')
Token: 'save' Etiqueta: 'VB' (Palabra Real, etiqueta Real): ('save', 'VB')
      'save' Etiqueta: 'NN' (Palabra Real, etiqueta Real): ('save', 'NN')
      'save' Etiqueta: 'IN' (Palabra Real, etiqueta Real): ('save', 'IN')
Token: '?' Etiqueta: '.' (Palabra Real, etiqueta Real): ('?', '.')
Token: 'happy' Etiqueta: 'JJ' (Palabra Real, etiqueta Real): ('happy', 'JJ')
Token: '...' Etiqueta: '.' (Palabra Real, etiqueta Real): ('...', '.')
Token: 'I' Etiqueta: 'PPSS' (Palabra Real, etiqueta Real): ('I', 'PPS')
Token: 'spies' Etiqueta: 'VBZ' (Palabra Real, etiqueta Real): ('spy', 'VB')
      'spies' Etiqueta: 'NNS' (Palabra Real, etiqueta Real): ('spy', 'NN')
Token: 'snagging' Etiqueta: 'VBG' (Palabra Real, etiqueta Real): ('snag', 'VB')
Token: 'moving' Etiqueta: 'VBG' (Palabra Real, etiqueta Real): ('move', 'VB')
      'moving' Etiqueta: 'NN' (Palabra Real, etiqueta Real): ('moving', 'NN')
      'moving' Etiqueta: 'JJ' (Palabra Real, etiqueta Real): ('moving', 'JJ')
Token: 'fishes' Etiqueta: 'VBZ' (Palabra Real, etiqueta Real): ('fish', 'VB')
      'fishes' Etiqueta: 'NNS' (Palabra Real, etiqueta Real): ('fish', 'NN')
Token: 'Sebastian' Etiqueta: 'NP' (Palabra Real, etiqueta Real): ('Sebastian', 'NP')
Token: 'visaing' Etiqueta: 'VBG' (Palabra Real, etiqueta Real): ('visa', 'VB')
Token: 'frolicking' Etiqueta: 'VBG' (Palabra Real, etiqueta Real): ('frolic', 'VB')
Token: 'she' Etiqueta: 'PPS' (Palabra Real, etiqueta Real): ('she', 'PPS')
Token: 'my' Etiqueta: 'PP5' (Palabra Real, etiqueta Real): ('my', 'PP5')
```

El analizador morfológico implementado tiene muchas bondades, ya que etiqueta los tokens con el estándar de las etiquetas *brown*, lo cual hace que su salida sea de igual forma estándar. Al igual que el tokenizador el diseño de este prototipo se utiliza un enfoque que no se encuentra documentado en ninguno de los documentos consultados.

En este capítulo se trata el proceso de análisis morfológico, que se encarga de etiquetar gramaticalmente las palabras apoyado en el diccionario. La siguiente etapa aprovechará estas etiquetas para realizar un análisis sintáctico y de acuerdo a las reglas gramaticales comprobar la correcta conformación de las frases e identificar la función sintáctica de cada una de las palabras.

7. ANÁLISIS SINTÁCTICO

7.1. INTRODUCCIÓN

En el análisis morfológico se generó un diccionario que contiene todos los tokens de entrada con las posibles etiquetas de cada uno. Este diccionario es el principal insumo para el análisis sintáctico. La tarea principal en este nivel es describir cómo las palabras de la oración se relacionan y cuál es la función que cada palabra realiza en esa oración, es decir, construir la estructura de la oración de un lenguaje.

En este capítulo se hablará de los dos enfoques usados para determinar las secuencias gramaticales de los idiomas y se hará uso de uno de estos enfoques para definir formalmente la gramática del idioma inglés para frases afirmativas en presente simple. El módulo aquí construido está apoyado principalmente en el *NLTK* y su analizador descendente, finalmente se mostrarán tres ejemplos que muestran las salidas del analizador (árbol sintáctico) a partir de información recogida del tokenizador (lista de tokens) y del analizador morfológico (diccionario de tokens).

Las normas o reglas para construir las oraciones se definen para los seres humanos en una forma **prescriptiva**, indicando las formas de las frases correctas y condenando las formas desviadas, es decir, indicando cuáles se prefieren en el lenguaje. En contraste, en el procesamiento lingüístico de textos, las reglas deben ser **descriptivas**, estableciendo métodos que definan las frases posibles e imposibles del lenguaje específico de que se trate¹²².

7.2. ASPECTOS TEÓRICOS

7.2.1. Enfoques de estructuras gramaticales

Las frases bien formadas son secuencias gramaticales que obedecen leyes gramaticales sin conocimiento del mundo, las no gramaticales deben postergarse a niveles que consideren la noción de contexto en un sentido amplio y el razonamiento. Establecer métodos que determinen únicamente las secuencias gramaticales en el procesamiento lingüístico de textos ha sido el objetivo de los

¹²² GALICIA HARO, Sofia Natalia. "Análisis sintáctico conducido por un diccionario de patrones de manejo sintáctico para lenguaje español." Tesis Doctoral. INSTITUTO POLITÉCNICO NACIONAL, 2000.

formalismos gramaticales en la **Lingüística Computacional**. En ella se han considerado dos enfoques para describir formalmente la gramaticalidad de las oraciones: **las dependencias** y **los constituyentes**.

7.2.1.1. Enfoque de dependencias

Este enfoque habla de las dependencias entre pares de palabras, donde una es principal o rectora y la otra está subordinada a (o dependiente de) la primera. Si cada palabra de la oración tiene una palabra propia rectora, la oración entera se ve como una estructura jerárquica de diferentes niveles, como un árbol de dependencias. La única palabra que no está subordinada a otra es la raíz del árbol.

La razón que ha motivado este enfoque es el sentido de las palabras. Por ejemplo, en la frase *young boys make jokes*¹²³, la palabra *young* es un modificador de atributo de la palabra *boys*, y *boys* es el sujeto de *make*.

En el enfoque de dependencias, la línea de trabajo más importante es la desarrollada por el investigador *Igor Mel'cuk*¹²⁴ desde los años sesenta, la **Meaning-Text Theory (MTT)**¹²⁵. Para *Mel'cuk*, en la sintaxis se describen los medios lingüísticos por los cuales se expresan todos los participantes que están implicados en el sentido mismo de los lexemas.

Bajo esta perspectiva, la descripción de conocimiento lingüístico es primordial. La descripción de los medios lingüísticos con los que se expresan los objetos del lexema se insertan junto con él en un diccionario, de esta forma se conoce de antemano cómo se relaciona el lexema con los distintos grupos de palabras en la oración. Por ejemplo, para el lexema *permanecer* aparecerá que utiliza la preposición *en* para introducir el lugar, que *solidaridad* utiliza la preposición *con*, y que el verbo *regalar* emplea un sustantivo para expresar el objeto donado y para introducir el receptor emplea la preposición *a*. Esto lleva a concluir que para hacer uso de esta teoría se requiere de un diccionario donde se establezcan todas las relaciones de dependencia entre los diferentes lexemas, por tanto, lograr una cobertura amplia de un lenguaje natural empleando estas relaciones requiere del establecimiento de una gran cantidad de conocimiento lingüístico que no se basa

¹²³ En español: los jóvenes hacen bromas.

¹²⁴ MEL'CUK, Igor. Profesor Departamento de Linguística y Traducción Universidad de Montreal (Rusia, 1932).

¹²⁵ Teoría del lenguaje que se basa en el hecho de que cualquier acto de comunicación lingüística implica un contenido (meaning), un signo oral o escrito (text) y una proyección (un conjunto de correspondencias entre significados y textos).

en lógica y que por lo tanto conlleva el enorme trabajo manual de la descripción de la colección completa de todos los posibles objetos de las palabras específicas.

7.2.1.2. Enfoque de constituyentes

Los constituyentes y la suposición de la estructura de frase, sugerida por *Leonard Bloomfield*¹²⁶ en 1933, es el enfoque donde las oraciones se analizan mediante un proceso de segmentación y clasificación. Se segmenta la oración en sus partes constituyentes, se clasifican estas partes como categorías gramaticales, después se repite el proceso para cada parte dividiéndola en subconstituyentes, y así sucesivamente hasta que las partes sean las partes de la palabra indivisibles dentro de la gramática (morfemas).

La suposición de frase y la noción de constituyente, se aplica de la siguiente forma. La frase *young boys make jokes* se divide en el grupo nominal *young boys* más el grupo verbal *makes jokes*, este último a su vez, se divide en el verbo *make* más el grupo nominal *jokes*.

En la perspectiva de constituyentes, la línea más importante de trabajo es la desarrollada por *Noam Chomsky*¹²⁷, desde los años cincuenta. *Chomsky* decía que lo que el ser humano sabe, cuando conoce un lenguaje, es un conjunto de palabras y reglas con las cuáles se pueden generar cadenas de esas palabras.

Bajo este enfoque, aunque existe un número finito de palabras en el lenguaje, es posible generar un número infinito de oraciones mediante esas reglas, que también se emplean para la comprensión del lenguaje. Aquí se deben tener en cuenta entonces dos cuestiones principales cuando se trata de la cobertura amplia de un lenguaje natural: el número de reglas y la definición concreta de ellas.

El número requerido de reglas para analizar las oraciones de un lenguaje natural no tiene límite predeterminado porque debe haber tantas reglas como sean requeridas para expresar todas las variantes posibles de las secuencias de palabras que los hablantes nativos pueden realizar. En cuanto a la definición, se generan mucho más secuencias de palabras de las que realmente quieren producirse. Por ejemplo, una regla para definir ***sintagmas nominales***¹²⁸ en inglés es: un *artículo indefinido*, seguido de un *sustantivo* y a continuación un *sintagma*

¹²⁶ BLOOMFIELD, Leonard. Antropólogo estadounidense, escritor de la obra *Lengua* (Chicago, 1887 – New Heaven, 1949).

¹²⁷ CHOMSKY, Avram Noam N. C. (Pensilvania, 1928) Op. Cit.

¹²⁸ Llámense así las frases no verbales encabezadas por un sustantivo. Por ejemplo, the beautiful girl (la hermosa niña).

preposicional. Sin embargo, esta regla define tanto *the table at the house*¹²⁹ como *the house at the shoe*¹³⁰ siendo ésta última una secuencia no gramatical.

Para este trabajo se ha adoptado el enfoque de constituyentes como esquema, debido a la alta complejidad y el alto nivel de conocimiento lingüístico que requiere la construcción de un **lexicón**¹³¹ que involucre relaciones de dependencias en caso de adoptarse el enfoque de dependencias.

Además de la estructura gramatical del texto, el análisis sintáctico también requiere la determinación de las funciones gramaticales de cada uno de los elementos terminales de la gramática (tokens), de tal forma que se puedan clasificar de acuerdo a su parte del habla (sustantivo, verbo, determinante, adjetivo, adverbio)¹³².

7.2.2. Parsing

Parsing es el término con el que se denomina el proceso de análisis sintáctico realizado en *PLN*. Consiste en un conjunto de operaciones algorítmicas necesarias para la comprensión sintáctica de una frase, generando como salida una representación arbórea del resultado.

Desde el punto de vista del **parser**¹³³, una gramática es un conjunto de reglas que describen cómo los distintos constituyentes se pueden combinar. Las combinaciones permitidas por la gramática son consideradas gramaticales, mientras que el resto son agramaticales. Formalmente, una lengua es un conjunto de oraciones, cada oración es una cadena de uno o más símbolos pertenecientes al vocabulario de la lengua. Desde esta perspectiva, una gramática no es más que una especificación formal y finita de este conjunto de oraciones. Esta especificación puede ser llevada a cabo por enumeración si el conjunto de oraciones es finito. Si el conjunto de oraciones es infinito o el número de posibles oraciones es demasiado elevado, resulta imposible o poco económico describirlas por enumeración. La otra manera es idear un mecanismo, es decir una gramática, capaz de decidir, mediante la aplicación de un determinado número de reglas, si determinada oración es o no es gramatical.

¹²⁹ En español: la mesa en la casa.

¹³⁰ En español: la casa en el zapato.

¹³¹ Se refiere a todos los lexemas que conforma un lenguaje, junto a su clasificación gramatical.

¹³² SAG, Ivan A. y WASOW, Tomas. Syntactic Theory: A Formal Introduction. Estados Unidos, enero de 1999.

¹³³ Nombre dado en inglés al analizador sintáctico.

Las operaciones de análisis que se realizan son agrupamientos parciales de los signos léxicos en unidades superiores. Por ejemplo, un *determinante* combinado con un *nombre* será tratado por una regla que permita la creación de un sintagma nominal. Esta unidad sintagmática se utilizará en otras reglas para formar unidades superiores. En el momento que se haya analizado la frase se tendrá conocimiento de todas las unidades léxicas que aparecen y la relación establecida entre ellas. El resultado de este análisis puede ser utilizado para realizar un análisis semántico.

El *parser* produce inferencias a partir de la información consultada en las reglas gramaticales. Sus algoritmos son considerados procedimientos de búsqueda que dirigen el proceso y elaboran la representación de la estructura de la frase. La función de un *parser* es, por tanto, identificar los elementos de la oración y especificar la relaciones entre ellos. Si una gramática puede generar un determinado número de oraciones, significa que el *parser* también puede reconocerlas como gramaticales para esa gramática.

Para poder analizar una frase es necesario encontrar una manera de generarla a partir de un símbolo inicial. Básicamente, esto puede hacerse de dos formas: análisis descendente y análisis ascendente.

7.2.2.1. Analizadores descendentes

Conocidos también con el nombre de *top-down*, se caracterizan por comenzar el proceso de análisis por las reglas más generales: si se considera la representación del análisis en forma arbórea, en la parte superior (*top*) aparecen las reglas más generales, y a medida que descendemos se procesan las más específicas, llegando finalmente a las unidades léxicas formadas por palabras. En modo *top-down* el analizador comienza por reglas generales como:

ORACION → SN SV

Esta regla indica que una oración está formada por un sintagma nominal seguido de un sintagma verbal. La siguiente regla a aplicar sería del tipo:

SN → ARTICULO SUSTANTIVO

Esta regla indica que un sintagma nominal está formado por un artículo, seguido de un sustantivo.

Otra regla básica es la correspondiente al sintagma verbal, donde se diga por ejemplo que puede estar compuesto sólo por un verbo o por un verbo más un sintagma nominal.

SV → VERBO | VERBO SN

En el proceso de **parsing** de la frase *the child cries* se aplicarían las siguientes reglas según este orden:

ORACION →	SN SV
SN →	ARTICULO SUSTANTIVO
ARTICULO →	'the'
SUSTANTIVO →	'child'
SV →	VERBO
VERBO →	'cries'

7.2.2.2. Análisis ascendente

Conocidos también como *bottom-up*, aplican las reglas en orden inverso al modelo anterior: el análisis comienza por la parte inferior del árbol (*bottom*), y se forman unidades complejas ascendiendo hacia la parte superior (*up*).

El primer paso es el reconocimiento de las categorías de las palabras:

ARTICULO →	'the'
SUSTANTIVO →	'child'
VERBO →	'cries'

Se forman posteriormente unidades sintagmáticas en forma ascendente:

SN →	ARTICULO SUSTANTIVO
SV →	VERBO

Los estados de análisis son generados por la asignación de una palabra a su categoría léxica correspondiente, o bien reemplazando la parte derecha de una regla por su parte izquierda.

La consideración esencial para la elección de uno de estos dos métodos es la de ramificación, normalmente se usará el método que permita una menor ramificación a partir de los nodos. Además, estos dos enfoques pueden combinarse, y de hecho es usual, en un solo *parser*. Este método se denomina *análisis ascendente con filtraje descendente* o también *análisis descendente con retrotrazado*. Para este trabajo se ha optado por utilizar el análisis descendente y más adelante se explicará la razón de esta escogencia.

7.3. MODELAMIENTO DEL ANALIZADOR SINTÁCTICO

El analizador sintáctico que se ha construido permite generar las diferentes estructuras gramaticales válidas que puede tener el texto de entrada. Para esto, se ha acogido el enfoque de constituyentes y dentro de este, se usa un *parser* descendente que permite escoger las diferentes oraciones bien formadas del texto. Se ha escogido adoptar el *parser* descendente pues para esta implementación el ascendente no serviría. Esto, debido a que el esquema de trabajo adoptado consiste en dar a los tokens todas sus posibles etiquetas para que el *parser* (descendente) detecte las oraciones bien formadas. Para usar el *parser* ascendente se necesitaría que cada token tuviera definida una única etiqueta, aspecto negativo que no permite su utilización en el prototipo aquí desarrollado.

La implementación de este módulo, al igual que la del tokenizador, hace uso del *NLTK*. Se utiliza módulo `parse.cfg` para definir la gramática con la cual se formarán las oraciones y del módulo `parse.rd` para realizar el análisis descendente de los tokens de entrada.

Las reglas de producción están conformadas por las producciones gramaticales y léxicas. Las producciones gramaticales definen la gramática de oraciones bien formadas del idioma inglés para frases afirmativas en presente simple. Las producciones léxicas indican las posibles etiquetas que puede tener cada uno de los tokens del texto de entrada, a partir de estas es que el *parser* define las posibles listas de producción de salida (árboles sintácticos), esto es, las posibles estructuras correspondientes a oraciones bien formadas que puede tener el texto de entrada.

Según esto, las producciones gramaticales se definen una sola vez porque siempre serán las mismas, mientras que las producciones léxicas varían dependiendo del texto de entrada.

Tabla 2 Gramática definida para el inglés, frases afirmativas, presente simple

FRASE →	ORACION ORACION CONJUNCION FRASE
ORACION →	SN SN SV
SN →	SUSTANTIVO NOMPROPIO ARTICULO SN ADJPOSESIVO SN ARTICULO SUSTANTIVO FP ADJETIVO SN PPS PPSS
SUSTANTIVO →	NN NNS
NOMPROPIO →	NP NP NOMPROPIO
ARTICULO →	AT
ADJPOSESIVO →	PP5
SV →	VERBO SN VERBO SN FP VERBO CONJUNCION VERBO VERBO FP VERBO
VERBO →	VB VBZ VBG TOBE TOBE VBG
CONJUNCION →	CC
FP →	PREPOSICION SN

PREPOSICION→	IN
ADJETIVO →	JJ JJ CONJUNCION JJ
TOBE →	BEM BER BEZ BEG

En la Tabla 2 se puede ver la estructura de las producciones gramaticales que se han definido para el analizador. Para esto, se ha partido de una construcción muy básica del idioma la cual dice que la oración es *un sujeto que realiza una acción sobre un objeto*. El sujeto se conoce como sintagma nominal, la acción que se realiza sobre el objeto se conoce como **sintagma verbal** y este a su vez está compuesto por una acción (parte verbal) realizada sobre un objeto (sintagma nominal). La definición anterior de composición del texto se define así:

- Una oración en su más mínima expresión puede ser una frase o una conjunción de frases.
- La frase está compuesta de una construcción nominal o sintagma nominal (SN) y una construcción verbal o sintagma verbal (SV). Puede considerarse también el caso de una frase compuesta solamente de un sintagma nominal. Para una definición más amplia de frase refiérase al Anexo E.
- Un sintagma nominal puede tener diferentes composiciones, cualquiera de ellas que le dan el carácter de sujeto.
- El sintagma verbal puede ser un solo verbo (no recae sobre un sujeto), un verbo más un sintagma nominal, u otras producciones que permitan indicar una acción.

7.4. INPLEMENTACIÓN DEL ANALIZADOR SINTÁCTICO

La clase `AnalizadorSintactico` tiene una función principal que define la gramática y con esta información instancia el *parser* descendente para determinar las diferentes producciones posibles de la gramática definida. Para definir las producciones léxicas se creó la función `generar_lexicon` quien a partir del diccionario genera una cadena con estas producciones.

La función realiza un recorrido al diccionario de tokens retornado por el analizador morfológico palabra por palabra (en este caso el recorrido es a través de los tipos de palabras que componen la frase). Luego, en el segundo `for`, por cada una de las palabras se recorre cada una de las etiquetas. Si el arreglo asociativo no contiene la etiqueta que está siendo analizada asigna la palabra a la cadena de texto que corresponde a la etiqueta. Sino se adiciona la palabra a esta cadena de texto, las palabras se separan con el caracter `'|'` ya que esta es la notación

utilizada por las gramáticas libres de contexto manejadas por el *NLTK*. Luego, se crea una variable tipo `str` que contendrá las producciones léxicas y mediante un recorrido en el arreglo asociativo se agrega la línea de texto que corresponde a la regla léxica para cada una de las etiquetas.

```
def _generar_lexicon(self):
    diccionarioTokens = {}
    for token in self.tokenizador.get_tokens():
        diccionarioTokens[token] = ''
    etiquetasLexicas = {}
    for palabra in diccionarioTokens.keys():
        for etiqueta in
            self.diccionarioMorfologico.get_categorias(palabra):
            if etiquetasLexicas.has_key(etiqueta):
                etiquetasLexicas[etiqueta] = etiquetasLexicas[etiqueta] +
                    " | '" + palabra + "'"
            else:
                etiquetasLexicas[etiqueta] = "'" + palabra + "'"
                lexicon = ""
        for etiqueta in etiquetasLexicas:
            lexicon = lexicon + "\n" + etiqueta + " -> " +
                etiquetasLexicas[etiqueta]
    return lexicon
```

A continuación se muestra el código fuente de la clase `AnalizadorSintactico`.

```
from traductor.librerias.DiccionarioInsensitive import KeyInsensitiveDict
from traductor.librerias.IO import Archivo
from traductor.tokenizacion import Tokenizador
from traductor.diccionario import Diccionario
from traductor.morfologico import AnalizadorMorfologico
from nltk_lite.parse.cfg import parse_grammar
from nltk_lite.parse.rd import RecursiveDescent
from os import environ
from os.path import sep

class AnalizadorSintactico:
    def __init__(self, objTokenizador, objDiccionarioMorfologico,
        archivoGramatica):
        if not environ.has_key('ARCHIVOS_TRADUCTOR'):
            raise IOError, 'No se encuentra la carpeta de archivos de texto.
                Defina la variable de entorno ARCHIVOS_TRADUCTOR'
        self.tokenizador = objTokenizador
        self.diccionarioMorfologico = objDiccionarioMorfologico
        self.archivoGramatica = Archivo(environ['ARCHIVOS_TRADUCTOR'] +
            sep + archivoGramatica, 'r')
        self.arbolesSintacticos = []

    def set_gramatica(archivoGramatica):
        self.archivoGramatica.set_nombre(environ['ARCHIVOS_TRADUCTOR'] +
```

```

        sep + archivoGramatica)

def analizar(self):
    produccionesGramaticales = self.archivoGramatica.leer()
    produccionesLexicas = self._generar_lexicon()
    gramatica = parse_grammar( produccionesGramaticales +
                               produccionesLexicas)
    parser = RecursiveDescent(gramatica)
    self.arbolesSintacticos =
        parser.get_parse_list(\
            self.tokenizador.get_tokens())

    return self.arbolesSintacticos

def _generar_lexicon(self):
    diccionarioTokens = {}
    for token in self.tokenizador.get_tokens():
        diccionarioTokens[token] = ''
    etiquetasLexicas = {}
    for palabra in diccionarioTokens.keys():
        for etiqueta in
            self.diccionarioMorfologico.get_categorias(palabra):
            if etiquetasLexicas.has_key(etiqueta):
                etiquetasLexicas[etiqueta] = etiquetasLexicas[etiqueta] +
                    " | '" + palabra + "'"
            else:
                etiquetasLexicas[etiqueta] = "'" + palabra + "'"
                lexicon = ""
        for etiqueta in etiquetasLexicas:
            lexicon = lexicon + "\n" + etiqueta + " -> " +
                etiquetasLexicas[etiqueta]

    return lexicon

def get_arboles(self):
    return self.arbolesSintacticos

```

Se debe importar la librería `DiccionarioInsensitive` para poder manejar las llaves independientemente de si están en mayúsculas o minúsculas, `Archivo` de `traductor.librerias.IO` para las operaciones de lectura y escritura de archivos, `environ` para hacer referencia a variables de entorno, `sep` que se refiere al separador que usa el sistema operativo para separar rutas. Además se importa la clase `Tokenizador`, `Diccionario` y `Morfologico`. Así como `parse_grammar` y `RecursiveDescent` para el análisis sintáctico.

```

from traductor.librerias.DiccionarioInsensitive import KeyInsensitiveDict
from traductor.librerias.IO import Archivo
from traductor.tokenizacion import Tokenizador
from traductor.diccionario import Diccionario
from traductor.morfologico import AnalizadorMorfologico
from nltk_lite.parse.cfg import parse_grammar

```

```
from nltk_lite.parse.rd import RecursiveDescent
from os import environ
from os.path import sep
```

Se define la clase y su constructor el cual instancia la clase `Tokenizador`, `Diccionario` y `AnalisisMorfologico`. Además de cargar el archivo de texto que tiene definida la gramática.

```
class AnalizadorSintactico:
    def __init__(self, objTokenizador, objDiccionarioMorfologico,
                 archivoGramatica):
        if not environ.has_key('ARCHIVOS_TRADUCTOR'):
            raise IOError, 'No se encuentra la carpeta de archivos de texto.
                           Defina la variable de entorno ARCHIVOS_TRADUCTOR'
        self.tokenizador = objTokenizador
        self.diccionarioMorfologico = objDiccionarioMorfologico
        self.archivoGramatica = Archivo(environ['ARCHIVOS_TRADUCTOR'] +
                                         sep + archivoGramatica, 'r')
        self.arbolesSintacticos = []
```

Se define la función `set_gramatica` para cambiar el nombre del archivo con las producciones gramaticales.

```
def set_gramatica(archivoGramatica):
    self.archivoGramatica.set_nombre(environ['ARCHIVOS_TRADUCTOR'] +
                                     sep + archivoGramatica)
```

Se define la función `analizar` que realiza la tarea de análisis sintáctico, en primer lugar se carga la generación del lexicón en la variable `lexicon`, y la gramática con la función `parse_grammar` del módulo `parse.cfg`¹³⁴, que se utiliza para definir gramáticas libres de contexto. Después de esto se almacena en la variable `parser` el resultado del análisis descendente obtenido con la función `RecursiveDescent` del módulo `parse.rd`¹³⁵ y se pasa a la variable `arbolSintactico` la lista de árboles sintácticos resultado del análisis descendente, pues el resultado de un análisis sintáctico pueden ser varios árboles.

```
def analizar(self):
    produccionesGramaticales = self.archivoGramatica.leer()
    produccionesLexicas = self._generar_lexicon()
    gramatica = parse_grammar( produccionesGramaticales +
                              produccionesLexicas)
    parser = RecursiveDescent(gramatica)
```

¹³⁴ La extensión `.cfg` (context free grammar) traduce al español: gramática libre de contexto

¹³⁵ La extensión `.rd` (recursive descent) traduce al español: análisis descendente recursivo

```

self.arbolesSintacticos =
    parser.get_parse_list(
        self.tokenizador.get_tokens())
return self.arbolesSintacticos

```

Por último además de la función `generar_lexicon`, se define la función `get_arboles` que retornar los árboles sintácticos generados en este análisis.

```

def get_arboles(self):
    return self.arbolesSintacticos

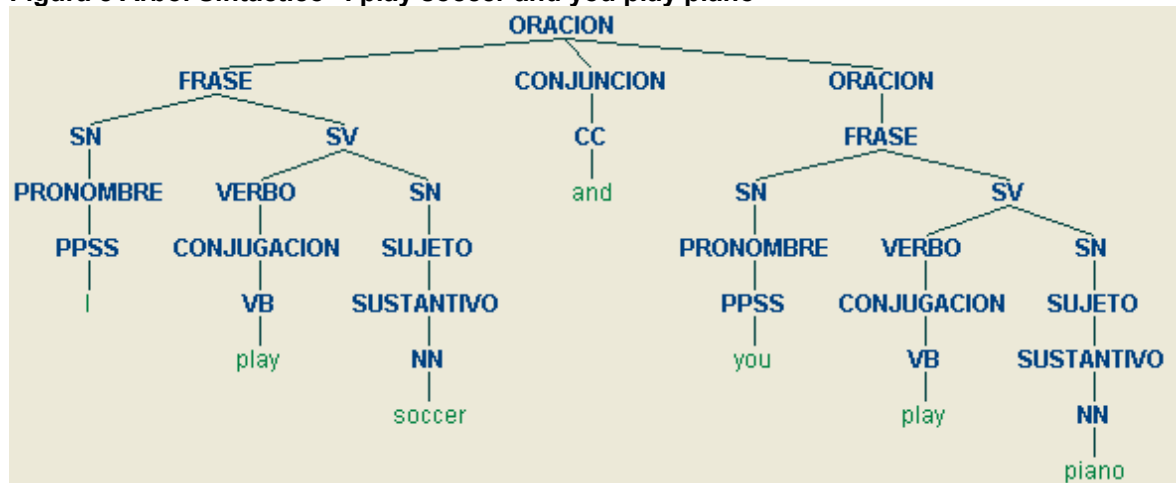
```

7.4.1. Ejemplos

A continuación se muestran tres ejemplos de aplicación del analizador sintáctico para tres frases afirmativas del inglés en presente simple:

- Ejemplo 1: I play soccer and you play piano

Figura 8 Árbol Sintáctico "I play soccer and you play piano"



Como se puede ver, la oración de entrada corresponde a una conjunción de dos frases, *I play soccer* y *you play piano*, mediante la conjunción *and*. Cada una de estas dos frases está formada por un sintagma nominal y un sintagma verbal, siendo el sintagma verbal un pronombre personal (sujeto) en ambos casos. El sintagma verbal de ambas oraciones está compuesto por un verbo en infinitivo (conjugación para primera y segunda persona del singular) el cual representa la acción realizada y un sintagma nominal que corresponde a un sustantivo e identifica al objeto de la acción.

La producción léxica generada para este caso es:

CC →	'and'
VB →	'play'
PPSS →	'I' 'you'
NN →	'play' 'soccer' 'piano'

Esta producción léxica se forma a partir de la salida del analizador morfológico, lo que se hace es agrupar por cada etiqueta los diferentes tokens asociados, así, como puede verse en la producción generada, *play* está en dos producciones diferentes, una donde figura como verbo y otra donde figura como sustantivo. Estas dos etiquetas fueron puestas por el analizador morfológico en la fase anterior. El parser identifica que la etiqueta correcta debe ser *VB* porque es con esa clasificación gramatical con la que la oración queda bien formada, es decir, si diera a *play* la clasificación de sustantivo (*NN*), el parser no generaría salidas válidas pues no encontraría un sintagma verbal para dejar completa cada oración.

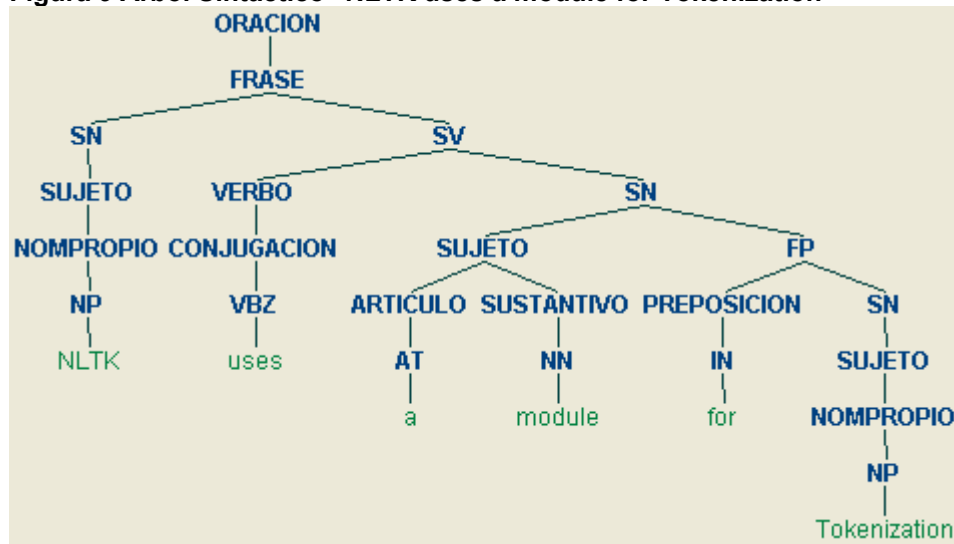
La salida del analizador en este caso es:

```
(ORACION:
(FRASE:
(SN: (PRONOMBRE: (PPSS: "I")))
(SV:
(VERBO: (CONJUGACION: (VB: "play")))
(SN: (SUJETO: (SUSTANTIVO: (NN: "soccer"))))))
(CONJUNCION: (CC: "and"))
(ORACION:
(FRASE:
(SN: (PRONOMBRE: (PPSS: "you")))
(SV:
(VERBO: (CONJUGACION: (VB: "play")))
(SN: (SUJETO: (SUSTANTIVO: (NN: "piano"))))))))
```

Esto corresponde al árbol de descomposición gramatical de la Figura 8.

- Ejemplo 2: NLTK uses a module for Tokenization

Figura 9 Árbol Sintáctico "NLTK uses a module for Tokenization"



La producción léxica generada para este caso es:

NN →	'module'
AT →	'a'
IN →	'for'
NP →	'Tokenization' 'NLTK'
VBZ →	'uses'
NNS →	'uses'

Salida del analizador:

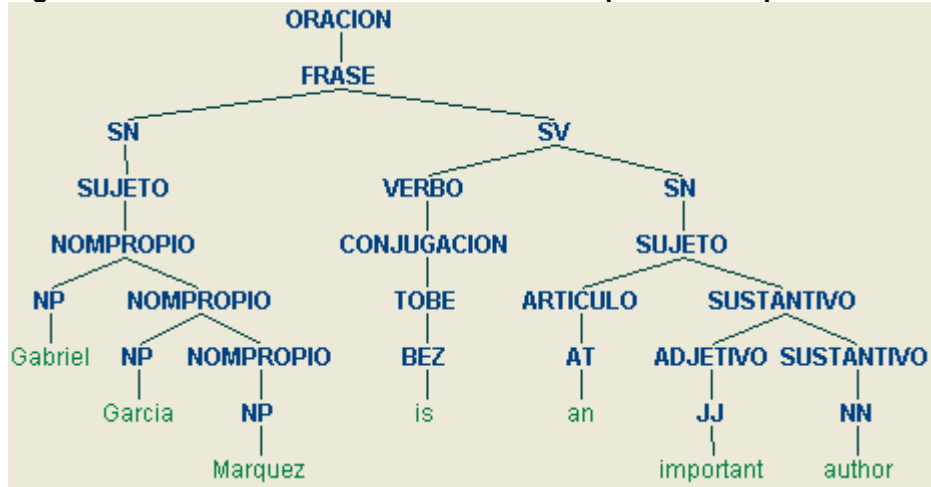
```

(ORACION:
  (FRASE:
    (SN: (SUJETO: (NOMPROPIO: (NP: "NLTK"))))
    (SV:
      (VERBO: (CONJUGACION: (VBZ: "uses")))
      (SN:
        (SUJETO:
          (ARTICULO: (AT: "a"))
          (SUSTANTIVO: (NN: "module")))
        (FP:
          (PREPOSICION: (IN: "for"))
          (SN: (SUJETO: (NOMPROPIO: (NP: "Tokenization"))))))))
  )
)
  
```

Esto corresponde al árbol de descomposición gramatical de la Figura 9.

- Ejemplo 3: Gabriel Garcia Marquez is an important author

Figura 10 Árbol Sintáctico "Gabriel Garcia Marquez is an important author"



La producción léxica generada para este caso es:

NP →	'Gabriel' 'Garcia' 'Marquez'
BEZ →	'is'
JJ →	'important'
AT →	'an'
NN →	'author'

Salida del analizador:

```

(ORACION:
  (FRASE:
    (SN:
      (SUJETO:
        (NOMPROPIO:
          (NP: "'Gabriel'")
          (NOMPROPIO:
            (NP: "'Garcia'")
            (NOMPROPIO: (NP: "'Marquez'"))))))
      (SV:
        (VERBO: (CONJUGACION: (TOBE: (BEZ: "'is'"))))
        (SN:
          (SUJETO:
            (ARTICULO: (AT: "'an'"))
            (SUSTANTIVO:
              (ADJETIVO: (JJ: "'important'"))
              (SUSTANTIVO: (NN: "'author'"))))))))
    )
  )
)
  
```

Esto corresponde al árbol de descomposición gramatical de la Figura 10.

En este capítulo se trató el proceso de análisis sintáctico usando el enfoque de constituyentes, se hizo uso del *parser* descendente del *NLTK* para la construcción

del prototipo en el cual se definió la gramática de frases afirmativas bien formadas en presente simple para el inglés, siendo su salida un árbol sintáctico que muestra la estructura gramatical del texto de entrada y sus correspondientes etiquetas. En el capítulo siguiente se abordará el análisis semántico, que usa el árbol sintáctico para definir un único género para cada token del mismo, lo cual permitirá que la síntesis al lenguaje destino se haga teniendo en cuenta el contexto temático de las oraciones, logrando finalmente que en la traducción se obtengan salidas inteligibles y no frases sin sentido.

8. ANÁLISIS SEMÁNTICO

8.1. INTRODUCCIÓN

En el capítulo anterior se trató el análisis sintáctico. En donde se describía cómo las palabras de la oración se relacionan y cuál es la función que cada palabra realiza en esa oración, es decir, construir la estructura de la oración de un lenguaje. Se trataron los dos enfoques usados para determinar las secuencias gramaticales de los idiomas y se hizo uso de del enfoque descendente para definir formalmente la gramática del idioma inglés para frases afirmativas en presente simple. Se mostró así mismo la funcionalidad con algunos ejemplos para los cuales se podía observar el árbol o estructura sintáctica.

Después de realizar el análisis sintáctico algunas ambigüedades pueden permanecer después de la aplicación de los métodos descritos anteriormente. La información semántica puede ser usada para resolver el problema. Aquí es donde se observa el por qué un buen diccionario debería tener información semántica de las palabras. Por ejemplo, las palabras relacionadas con música deberían estar marcadas como tal.

En el presente capítulo se hablará del **análisis semántico**. Aunque es una de las etapas más importantes, ya que es el análisis semántico el encargado de tomar decisiones para que las frases que sintácticamente son correctas, a luz de la semántica pueden tener diferente interpretación de acuerdo al contexto, no es muy común observar que los traductores empleen información semántica para realizar el proceso de traducción.

8.2. ASPECTOS TEÓRICOS

8.2.1. Representación del conocimiento

De los diferentes análisis que se han tratado en el presente documento, la sintaxis ha sido durante mucho tiempo y aún sigue siendo el nivel al que la lingüística le ha prestado mayor atención. Esto se puede justificar por dos razones principales en cuanto al tratamiento automático del lenguaje natural según *Rich E y Knight K.*¹³⁶

El procesamiento semántico funciona sobre los constituyentes de la oración. Si no existe un paso de análisis sintáctico, el sistema semántico debe identificar sus propios

¹³⁶ RICH E. y KNIGHT, K. (1991) Introduction to Artificial Networks. Mac Graw-Hill.

constituyentes. Por otro lado, si se realiza un análisis sintáctico, se restringe enormemente el número de constituyentes a considerar por el semántico, mucho más complejo y menos fiable. El análisis sintáctico es mucho menos costoso computacionalmente hablando que el análisis semántico (que requiere inferencias importantes). Por tanto, la existencia de un análisis sintáctico conlleva un considerable ahorro de recursos y una disminución de la complejidad del sistema.

Aunque frecuentemente se puede extraer el significado de una oración sin usar hechos gramaticales, no siempre es posible hacerlo.

Para realizar análisis semántico se debe tener en cuenta que existen tres tipos de representación del conocimiento:

- **Conocimiento semántico:** Conocimiento lingüístico el cual es independiente del contexto.
- **Conocimiento pragmático:** Es el tipo de conocimiento lingüístico el cual se relaciona con el contexto. Es el estudio del modo en que el contexto influye en la interpretación del significado.
- **Conocimiento del mundo real:** En general se refiere al sentido común, es un conocimiento no lingüístico acerca del mundo real.

Es importante dejar sentado que la distinción entre los tres no siempre es muy clara y muchas veces se puede llegar a dudar sobre cual de estas distinciones es la que se aplica en un momento determinado.

8.2.1.1. Conocimiento Semántico

La semántica tiene relación con el significado de las palabras y cómo estas se combinan para dar significados a las frases. La manera de realizar un análisis que tenga en cuenta este tipo de conocimiento es en primer lugar a través de los significados de las palabras y más adelante los significados de las frases hasta llegar al nivel de oraciones.

Hay muchas maneras en las cuales se puede representar el conocimiento, pero una de las que han resultado ser más útiles en el campo de la TA envuelve el asociar las palabras con características semánticas las cuales corresponden a sus componentes de sentido.

El asociar a las palabras características semánticas puede resultar muy útil porque algunas palabras colocan restricciones en las cuales otra clase de palabras pueden o no ocurrir. Por ejemplo para la palabra en inglés *play* actuando como verbo puede significar *jugar, tocar (interpretar un instrumento musical) o actuar*. De

esta manera en la frase *I play piano* la palabra *piano* obliga a que el verbo *play* tenga el significado asociado con la ejecución de una acción que implique un instrumento musical en este caso *tocar*. Quedando la frase traducida como *yo toco piano*.

8.2.1.2. Conocimiento pragmático

Como ya se dijo existe distinción entre la semántica o significado independiente del contexto y la pragmática o significado dependiente del contexto. El termino contexto se usa ambiguamente para referirse al resto del texto en el cual la oración puede ocurrir (algunas veces es llamado discurso) y a las circunstancias externas del texto por si mismo, tal como quién es el autor del texto y el entorno social en el cual ocurre, los cuales podrían contribuir a su interpretación. Pero para el análisis a realizar se va a considerar la primera acepción.

Para observar la importancia del discurso se consideran los pronombres anafóricos, los cuales hacen referencia a un antecedente en el texto. Por ejemplo la palabra *it* en inglés en la frase *Sam took the cake from the table. Then he ate it.*¹³⁷ Hace referencia a la palabra *cake*.

Es fácil identificar que *it* hace referencia a un sustantivo en singular que ha sido previamente expuesto en el texto. También puede asumirse que hace referencia a un sustantivo de la frase inmediatamente anterior (esto si se asume que estás son las primeras frases del texto). En el ejemplo se puede observar que *it* puede hacer referencia a tres sustantivos: *Sam*, *cake* y *table*. Teniendo en cuenta las restricciones de número y género *it* no puede referirse a *Sam*, aún así queda escoger entre *cake* o *table*.

Es claro que la interpretación del conocimiento pragmático puede llegar a depender de muchas variables entre ellas la intención del autor (En el sentido de que quiere dar a entender). Este tipo de conocimiento es mucho más difícil de analizar que el pasado porque tiene en cuenta el contexto en donde se desarrolla la situación descrita en la oración.

8.2.1.3. Conocimiento del mundo real

Fácilmente se podría ingerir que todo el conocimiento necesario para extraer el significado de un texto y traducirlo puede ser obtenido del texto y su contexto. Sin embargo, esto no es del todo cierto y puede ilustrarse claramente con la oración

¹³⁷ Esta frase puede ser traducida como El tomó el pastel de la mesa y luego se lo comió, el pronombre anafórico para esta frase en español es la palabra *se*.

The pen is in the box and the box is in the pen. La cual debería ser traducida como *el lapicero está en la caja y la caja está en la cuna*. El problema asociado a esta frase está en la palabra *pen* la cual tiene dos significados *lapicero* y *cuna*. En la primera frase de la oración *pen* es *lapicero* y en la segunda es *cuna*. Esto se puede ingerir porque no es posible que una cuna quepa dentro de un lapicero. Por lo menos no en el mundo real y aunque esto puede resultar obvio para una persona para un sistema de TA deducir algo semejante es muy complicado.

El conocimiento real como se puede observar en el sencillo ejemplo acabado de exponer envuelve un factor que si bien el ser humano posee, las máquinas aún no y es el razonamiento dado por el sentido común. Actualmente la manipulación y representación de tal conocimiento es una de las líneas de investigación más destacadas y es una de las razones de ser de una disciplina completa, la **Artificial Intelligence (AI)**¹³⁸. Los problemas de representación y manipulación del conocimiento lingüístico se tornan insignificantes comparados con los problemas de representación del mundo real. Existen dos problemas asociados a la representación del conocimiento.

En primer lugar, se debe tener en cuenta que tal conocimiento generalmente debe estar sujeto a revisión y aún así no es garantizado que sea correcto. Si se tiene en cuenta que incluso los seres humanos en ocasiones tienen problemas para asumir algo que tiene cierta contrariedad, pueden manejar supuestos (tal como el ejemplo de la cuna y el lapicero). Sin embargo, esto es extremadamente difícil de automatizar.

Como segundo, la gran cantidad de tal conocimiento que se debe tener (tal como tamaños relativos de cada cosa, por ejemplo) Aún así, existen algunos métodos de representación que son útiles para algunas clases de conocimiento.

8.3. MODELAMIENTO DEL ANALIZADOR SEMÁNTICO

Como se ha hecho notar, el análisis semántico es uno de los pasos más complejos en el proceso de Traducción Automática. Requiere una gran capacidad computacional así como algoritmos de alta complejidad. Tanto así, que actualmente es uno de los temas en los cuales se han centrado los esfuerzos últimamente. No obstante los resultados son pocos debido precisamente a dicha complejidad.

¹³⁸ IA En español Inteligencia Artificial, se define como aquella inteligencia exhibida por artefactos creados por humanos (es decir, artificial). A menudo se aplica hipotéticamente a los computadores.

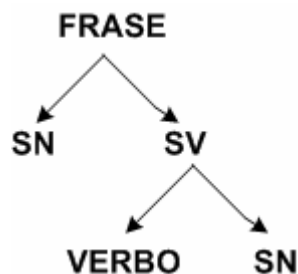
Si bien, en el presente documento se ha querido abordar el análisis semántico de una manera poco profunda, en el sentido que se ha centrado en resolver los problemas de orden de conocimiento semántico, no teniendo en cuenta el conocimiento de tipo pragmático y del mundo real. Aún así, es importante hacer énfasis en la dificultad asociada al proceso. Tanto así, que la mayoría de los sistemas de traducción automática optan por no realizar un proceso de análisis semántico.

Teniendo en cuenta que el prototipo desarrollado en el presente trabajo toma como espectro las frases en presente simple, el análisis se enfoca a resolver problemas de tipo de conocimiento semántico en frases de presente simple. De esta manera se intentará elaborar un módulo de análisis semántico que permita asociar las características semánticas existentes entre las palabras que componen una frase, algo que no es muy común y que tiene un alto grado de dificultad. Para dar un ejemplo claro en la frase *i play the piano* que debe ser traducida correctamente como *yo toco el piano* es necesario identificar que la palabra *play* corresponde al verbo *tocar* en español y no *jugar*. Esto teniendo en cuenta la relación existente entre los roles musicales existentes entre *piano* y *play*.

Aunque a simple vista el proceso parece sencillo y alguien podría decir que basta simplemente con comparar la relación existente entre palabras adyacentes, no es correcto pensar esto, ya que una palabra adyacente a otra no necesariamente está relacionada en la misma frase. Puede ocurrir de igual manera que las palabras que se relacionan no sean adyacentes tal como en el ejemplo anterior en donde el artículo *the* separa el verbo *play* del sustantivo *piano*.

Para abordar el problema, se hace necesario retomar la construcción básica del idioma la cual dice que la frase es *un sujeto que realiza una acción sobre un objeto*. El sujeto se conoce como sintagma nominal, la acción que se realiza sobre el objeto se conoce como sintagma verbal y este a su vez está compuesto por una acción (parte verbal) realizada sobre un objeto (sintagma nominal). De tal manera, un árbol que modela la anterior regla gramatical se puede observar en la Figura 11.

Figura 11 Estructura gramatical básica de una frase



Teniendo en cuenta lo anterior, es necesario encontrar la manera de relacionar los roles existentes en cada una de las partes en que se constituye la frase, el sintagma nominal, la parte del verbo del sintagma verbal y el sintagma nominal del sintagma verbal. Se debe tener en cuenta que este análisis es recursivo ya que no es posible determinar la profundidad del árbol que representa la frase. El análisis debe ser efectuado al nivel de la frase y debe ser capaz de relacionar adecuadamente la información de restricciones selectivas.

Dadas las anteriores especificaciones, se definen los siguientes conjuntos:

- $\text{rolesSN} = \{ x \mid x \text{ es un rol de la palabra } w \text{ en la categoría } cat, \text{ tal que } w \text{ pertenece al sintagma nominal de la frase actuando con la categoría } cat \}$

Por ejemplo para el sintagma nominal *the football*. El artículo *the* no cuenta con roles. Mientras que *football* quien actúa como sustantivo cuenta con el rol *Dep*. De esta forma, el conjunto rolesSN se compone de la siguiente manera:

$\text{rolesSN} = \{\text{NOROL}, \text{Dep}\}$

NOROL o rol por defecto, aparece en el conjunto para conservar la estructura del algoritmo en caso de que ninguna palabra posea roles específicos, también, para que en el momento de relacionar los roles y no exista una relación entre ellos se cree entonces una relación de roles por defecto. Esto se hace porque para poder acceder a un significado en el diccionario es necesario especificar un rol aunque sea el rol por defecto.

- $\text{rolesSVVerbo} = \{ x \mid x \text{ es un rol de la palabra } w \text{ en la categoría } cat, \text{ tal que } w \text{ pertenece a la parte del verbo del sintagma verbal de la frase actuando con la categoría } cat \}$

Por ejemplo para el sintagma verbal *play soccer*. *Play* quien actúa como verbo cuenta con los roles *Mús*, *Teat*, *Dep*. De esta forma, el conjunto rolesSVSN se compone de la siguiente manera:

$\text{rolesSVVerbo} = \{\text{NOROL}, \text{Mús}, \text{Teat}, \text{Dep}\}$

- $\text{rolesSVSN} = \{ x \mid x \text{ es un rol de la palabra } w \text{ en la categoría } cat, \text{ tal que } w \text{ pertenece a la parte del sintagma nominal del sintagma verbal de la frase actuando con la categoría } cat \}$

Por ejemplo para la frase *i play the beautiful piano*. El sintagma nominal del sintagma verbal es *the beautiful piano*. El artículo *the* y el adjetivo *beautiful* no

cuentan con roles. Mientras que *piano* quien actúa como sustantivo cuenta con el rol *Mús.* De esta forma, el conjunto rolesSVSN se compone de la siguiente manera:

$\text{rolesSVSN} = \{\text{NOROL}, \text{Mús}\}$

Posteriormente, se debe definir una función que se encargue de analizar los roles comunes entre los componentes de la oración. Dicha función debe ser capaz de analizar la relación existente entre el sintagma nominal y el sintagma verbal de una frase, teniendo en cuenta también los componentes de este último. La función puede constituirse entonces como un motor de inferencia encargado de realizar dicho procedimiento.

El motor de inferencia descrito puede ser tan complejo como se quiera, dadas las características del problema es posible incluso llegar a utilizar algoritmos de inteligencia artificial, así como algoritmos desarrollados en lenguajes de programación declarativos como ProLog, LISP, Haskell los cuales tienen más ventajas para el manejo de la alta recursividad que el problema presenta.

En términos generales el motor de inferencia debe ser capaz de realizar lo siguiente:

```
rolesSN', rolesSVVerbo', rolesSVSN' = analizar_roles_comunes(rolesSN,
rolesSVVerbo, rolesSVSN)
```

En donde rolesSN' es el conjunto de roles que deben ser aplicados al sintagma nominal y que ya, habiendo hecho uso del motor de inferencia, se han relacionado adecuadamente con los roles del resto de componentes de la frase. De igual forma rolesSVVerbo' y rolesSVSN' son los conjuntos de roles que deben ser aplicados a la parte del verbo del sintagma verbal y al sintagma nominal del sintagma verbal respectivamente, teniendo en cuenta también, que similarmente, habiendo hecho uso del motor de inferencia, se han relacionado adecuadamente con los roles del resto de componentes de la frase.

Para finalizar, se realiza una leve descripción del motor de inferencia usado en el prototipo del presente trabajo. En principio, se tienen funciones que se encargan de descender recursivamente en el árbol que representa la frase para asignar a las variables rolesSN, rolesSVVerbo y rolesSVSN, los conjuntos de roles definidos anteriormente.

Posteriormente haciendo uso de una intersección entre rolesSVVerbo y rolesSVSN se encuentran los roles comunes en el sintagma verbal. En caso de haber roles en común, estos últimos se interceptan con rolesSN obteniendo de

esta forma los roles comunes en la frase, si los hay, a cada uno de los conjuntos de roles de los componentes de la frase se les asigna dicho valor. Ahora bien, si no hay roles comunes, a rolesSN' se le asigna el valor de rol por defecto y los roles de los componentes del sintagma verbal se igualan a la intercepción de los componentes del sintagma verbal. Por último si la intercepción entre los componentes del sintagma verbal no produce roles en común se realiza una última intercepción entre el sintagma nominal de la frase y el componente del verbo del sintagma verbal en caso de haber roles en común rolesSN' y rolesSVVerbo' se igualan al valor obtenido en dicha intercepción.

Con el motor de inferencia explicado previamente se resuelven varios problemas de conocimiento de tipo semántico en el proceso de traducción automática. Aún así, debe tenerse en cuenta que dicho desarrollo marca un comienzo de lo que puede ser un motor de inferencia mucho más complejo, en donde las relaciones no se realicen por la intercepción de roles en común, sino que se pueda realizar un análisis mucho más profundo de dichos roles. En la siguiente sección se describe detalladamente la implementación del prototipo de análisis semántico, mostrando de manera práctica y sencilla su funcionalidad y aplicabilidad del modelo descrito.

8.4. IMPLEMENTACIÓN DEL ANALIZADOR SEMÁNTICO

Si bien en el presente documento se hace referencia a los diferentes tipos de representación del conocimiento, la implementación de estos en el prototipo no se ha llevado a cabo. En primer lugar por la complejidad del problema y en segundo lugar porque con lo desarrollado en el prototipo el lector puede hacerse a una idea de la importancia de realizar análisis semántico. De igual manera se muestra la funcionalidad de un sencillo motor de inferencia de conocimiento semántico el cual es capaz de eliminar ambigüedades de orden lingüístico.

Este motor de inferencia es capaz de asociar las clasificaciones comunes entre los componentes de una oración siendo posible resolver problemas de orden semántico como el que se presenta en la frase *i play soccer and she plays piano*. En el ejemplo anterior el verbo *play* puede ser asociado al objeto sobre el cual actúa (en este caso *piano* o *soccer*). Para la frase *The kid bleats and the kid speaks* es capaz de identificar la clasificación para *kid*, el cual en la primera frase es *cabrito* y en la segunda es *niño*. Esto se logra, haciendo uso del árbol sintáctico retornado por el análisis sintáctico, la información de carácter semántico que está incluida en el diccionario y una función que examina la relación existente entre los componentes de la frase.

Se define la clase `AnalizadorSemantico` cuyo constructor instancia la clases `Tokenizador`, `Diccionario`, `AnalizadorMorfologico` y `AnalizadorSintactico`.

```
class AnalizadorSemantico:
    def __init__(self, objTokenizador, objDiccionario,
                  objDiccionarioMorfologico, objSintactico):
        self.tokenizador = objTokenizador
        self.diccionario = objDiccionario
        self.diccionarioMorfologico = objDiccionarioMorfologico
        self.sintactico = objSintactico
        self.arbolesSemanticos = []
```

La función `analizar` realiza el proceso de análisis semántico.

```
def analizar(self):
    self.arbolesSemanticos = []
    for arbolSintactico in self.sintactico.get_arboles():
        arbolSemanticoAmbiguo = self._agregar_rol_semanticos(
                                arbolSintactico)
        arbolSemanticoNOAmbiguo = self._eliminar_ambigüedades_semanticas(
                                arbolSemanticoAmbiguo)
        self.arbolesSemanticos.append(arbolSemanticoNOAmbiguo)
    return self.arbolesSemanticos
```

A continuación se da una explicación a la función `analizar`.

```
def analizar(self):
```

El proceso se realiza para cada uno de los árboles sintácticos recibidos en la lista de árboles sintácticos.

```
self.arbolesSemanticos = []
for arbolSintactico in self.sintactico.get_arboles():
```

Se crea una variable `arbolSemanticoAmbiguo` el cual es resultado de agregar los roles semánticos al árbol sintáctico.

```
arbolSemanticoAmbiguo = self._agregar_rol_semanticos(
                        arbolSintactico)
```

`arbolSemanticoNOAmbiguo` tiene el valor retornado por la función `eliminar_ambigüedades_semanticas` produciendo un árbol semántico libre de ambigüedades de orden semánticas.

```
arbolSemanticoNOAmbiguo =self._eliminar_ambigüedades_semanticas(  
    arbolSemanticoAmbiguo)
```

Se agrega a la lista `listaArbolesSemanticos` el árbol con el análisis semántico realizado.

```
self.arbolesSemanticos.append(arbolSemanticoNOAmbiguo)
```

Finalmente se retorna la lista de árboles semánticos.

```
return self.arbolesSemanticos
```

Se define la función `eliminar_ambigüedades_semanticas` la cual elimina ambigüedades en las etiquetas semánticas de un árbol sintáctico. Recibe el árbol sintáctico al cual ya se le han agregado previamente las etiquetas semánticas.

Desciende recursivamente ubicando árboles que contienen etiquetas de *'FRASE'*, al encontrar una frase realiza una clasificación del sintagma nominal y el sintagma verbal. Posteriormente hace un análisis a estos, relacionando las etiquetas comunes entre los dos. Finalmente reetiqueta los roles teniendo en cuenta solo los retornados por el análisis realizado. Retorna el árbol sintáctico enviado, eliminando de este las etiquetas asociadas a cada palabra no validas semánticamente para la estructura de la frase.

```
def _eliminar_ambigüedades_semanticas(self, arbolSintactico):  
    if arbolSintactico.node == 'ORACION':  
        resultadoEtiquetacion = map(  
            self._eliminar_ambigüedades_semanticas,  
            arbolSintactico)  
        arbolSintactico = Tree(arbolSintactico.node,  
                               resultadoEtiquetacion)  
    else arbolSintactico.node == 'FRASE':  
        arbolSintacticoSN = arbolSintactico[0]  
        if len(arbolSintactico) == 2:  
            rolesSN = self._clasificar_SN(arbolSintacticoSN)  
            arbolSintacticoSV = arbolSintactico[1]  
            rolesSV = self._clasificar_SV(arbolSintacticoSV)  
            (rolesSN, rolesSV) = self._analizarEtiquetasComunes(rolesSN,  
                                                                    rolesSV)  
  
            arbolSintacticoSN = self._asignar_categorias_SN(  
                arbolSintacticoSN, rolesSN)  
            arbolSintacticoSV = self._asignar_categorias_SV(  
                arbolSintacticoSV, rolesSV)  
            arbolSintactico = Tree(arbolSintactico.node,  
                                   [arbolSintacticoSN, arbolSintacticoSV])  
        else:
```

```

        arbolSintacticoSN = self._asignar_categorias_SN(
            arbolSintacticoSN, ['NOROL'])
        arbolSintactico = Tree(arbolSintactico.node,
                               [arbolSintacticoSN])
    return arbolSintactico

```

A continuación se ofrece una explicación de la función `eliminar_ambigüedades_semanticas`.

```

def _eliminar_ambigüedades_semanticas(self, arbolSintactico):

```

Si el árbol recibido es una oración, significa que no se ha llegado al nivel de oración que es donde se realiza el proceso, así pues se debe descender recursivamente hasta encontrar frases.

```

    if arbolSintactico.node == 'ORACION':

```

Se utiliza un mapeo para realizar la recursividad, a cada uno de los hijos de `arbolSintactico` se le aplica la función `eliminar_ambigüedades_semanticas`, este mapeo da como resultado una lista (la cual es una lista de hijos a los cuales ya se les ha aplicado la función `eliminar_ambigüedades_semanticas` (Es decir ya se han eliminado las ambigüedades semánticas).

```

        resultadoEtiquetacion = map(
            self._eliminar_ambigüedades_semanticas,
            arbolSintactico)

```

Se asigna al `arbolSintactico` un nuevo árbol sintáctico compuesto por los hijos ya etiquetados semánticamente.

```

        arbolSintactico = Tree(arbolSintactico.node,
                               resultadoEtiquetacion)

```

Si el árbol es una frase se realiza una clasificación del sintagma nominal y el sintagma verbal. Para esto se hace uso de la funciones `clasificar_SN` y `clasificar_SV`. Posteriormente hace un análisis a estos, relacionando las etiquetas comunes entre los dos. La función `analizarEtiquetasComunes` es la encargada de realizar este proceso. Finalmente reetiqueta los roles teniendo en cuenta solo los retornados por el análisis realizado. A través de las funciones `asignar_etiquetas_SN` y `asignar_etiquetas_SV`. Por último retorna el nuevo árbol reetiquetado y sin ambigüedades en los roles.

```

else arbolSintactico.node == 'FRASE':
    arbolSintacticoSN = arbolSintactico[0]
    if len(arbolSintactico) == 2:
        rolesSN = self._clasificar_SN(arbolSintacticoSN)
        arbolSintacticoSV = arbolSintactico[1]
        rolesSV = self._clasificar_SV(arbolSintacticoSV)
        (rolesSN, rolesSV) = self._analizarEtiquetasComunes(rolesSN,
                                                             rolesSV)

        arbolSintacticoSN = self._asignar_categorias_SN(
            arbolSintacticoSN, rolesSN)
        arbolSintacticoSV = self._asignar_categorias_SV(
            arbolSintacticoSV, rolesSV)
        arbolSintactico = Tree(arbolSintactico.node,
                               [arbolSintacticoSN, arbolSintacticoSV])
    else:
        arbolSintacticoSN = self._asignar_categorias_SN(
            arbolSintacticoSN, ['NOROL'])
        arbolSintactico = Tree(arbolSintactico.node,
                               [arbolSintacticoSN])

return arbolSintactico

```

Las funciones que se han mencionado anteriormente: `clasificar_SN`, `clasificar_SV`, `analizarEtiquetasComunes`, `asignar_etiquetas_SN` y `asignar_etiquetas_SV` han sido completamente desarrolladas en el prototipo. Su código fuente no se ha incluido en el documento debido a su gran extensión y alta complejidad. Aún así se dará una breve explicación de su funcionamiento y si se quiere ahondar más en estas se aconseja revisar el código fuente, en el cual las funciones han sido ampliamente documentadas.

`clasificar_SN`: Realiza una clasificación de los diferentes roles para un árbol de tipo SN (Sintagma nominal).

Entrada: árbol sintáctico que corresponde a un SN (Sintagma nominal)

Proceso: usa la función `generos_arbol` el cual retorna todos los roles que existen en las hojas de un árbol para clasificar el sintagma nominal.

Salida: lista de roles para el árbol correspondiente al sintagma nominal recibido.

`clasificar_SV`: Realiza una clasificación de los diferentes roles para los componentes del SV (Sintagma verbal).

Entrada: árbol sintáctico que corresponde a un SV (Sintagma verbal)

Proceso: Identifica la estructura del árbol y asigna los roles para cada uno de los componentes del sintagma verbal.

Salida: Retorna una tupla de la forma `(rolesSVVerbo, rolesSVSustantivo)` que corresponde a los roles del componente verbal y componente nominal del sintagma verbal recibido.

Analizar_etiquetas_comunes: Realiza el análisis entre los roles de un SN el componente V de un SV y el componente SN de un SV. Es importante resaltar que es esta función la encargada de realizar la ingerencia acerca de que tipos de roles se deben seleccionar.

Entrada: lista de roles del SN y tupla `rolesSV` compuesta por (`rolesSVVerbo`, `rolesSVSSN`) que corresponden a la parte verbal del sintagma Verbal y la parte del sintagma nominal del sintagma verbal respectivamente.

Proceso: realiza un análisis entre las listas recibidas para realizar un proceso de selección de roles. Así, se eliminan ambigüedades en los roles y se obtienen significados que han sido asociados los diferentes sintagmas en la oración.

Salida: Se retorna una nueva tupla compuesta por la lista de roles del SN y la tupla `rolesSV` compuesta por (`rolesSVVerbo`, `rolesSVSSN`) que corresponden a la parte verbal del sintagma Verbal y la parte del sintagma nominal del sintagma verbal respectivamente. Estas variables se les ha realizado un proceso de análisis y se han depurado para guardar coherencia las unas con las otras, sino hay coherencia se retornan listas vacías.

asignar_etiquetas_SN: Asigna a cada hoja de un árbol sintáctico tipo SN las etiquetas comunes con la lista de roles recibidos.

Entrada: Árbol sintáctico SN a realizar el proceso y lista de roles para comparar con la lista de roles de cada una de las palabras que componen el árbol.

Proceso: Desciende recursivamente y compara los roles de cada una de las hojas del árbol y asigna como nuevos roles los roles comunes entre las que existían y los recibidos como parámetro.

Salida: Retorna un nuevo árbol con los roles de cada una de las hojas del árbol conteniendo como nuevos roles los roles comunes entre las que existían y los recibidos como parámetro.

asignar_etiquetas_SV: Asigna a cada hoja de un árbol sintáctico tipo SV las etiquetas comunes con la lista de roles recibidos.

Entrada: Árbol sintáctico SV a realizar el proceso y lista de roles para comparar con la lista de roles de cada una de las palabras que componen el árbol.

Proceso: Desciende recursivamente y compara los roles de cada una de las hojas del árbol y asigna como nuevos roles los roles comunes entre las que existían y los recibidos como parámetro.

Salida: Retorna un nuevo árbol con los roles de cada una de las hojas del árbol conteniendo como nuevos roles los roles comunes entre las que existían y los recibidos como parámetro.

8.4.1. Funcionamiento

Se ha probado la funcionalidad del módulo descrito anteriormente, con diversos ejemplos. Entre ellos los siguientes:

I play soccer and you play piano
I play soccer and she plays the pianos
Sebastian works with Tokenization
I'm working with Tokenization
Jose Ferney Franco fishes in the river
The young boy makes jokes
I button the button
The computer is an electronic device
The kid plays with the pianos and the girl plays with dolls
The kid bleats and the kid speaks
The kid plays piano
The kid plays with the girl
I work tokenization with Sebastian
The kid plays a match with the girl and the kid plays a piano with the girl
New York is a city

A continuación, se muestra el resultado de ejecutar análisis morfológico a dos de las frases anteriores: I play soccer and she plays the pianos y The kid bleats and the kid speaks. Cuya traducción correcta es *Yo juego fútbol y ella toca los pianos y El cabrito bala y el niño habla*.

En los ejemplos anteriores existen ambigüedades semánticas al tratar de traducir el texto, en el primer ejemplo el verbo *play* puede ser tres cosas (*tocar, jugar o actuar*) y para el segundo ejemplo el sustantivo *kid* puede ser *niño* o *cabrito*.

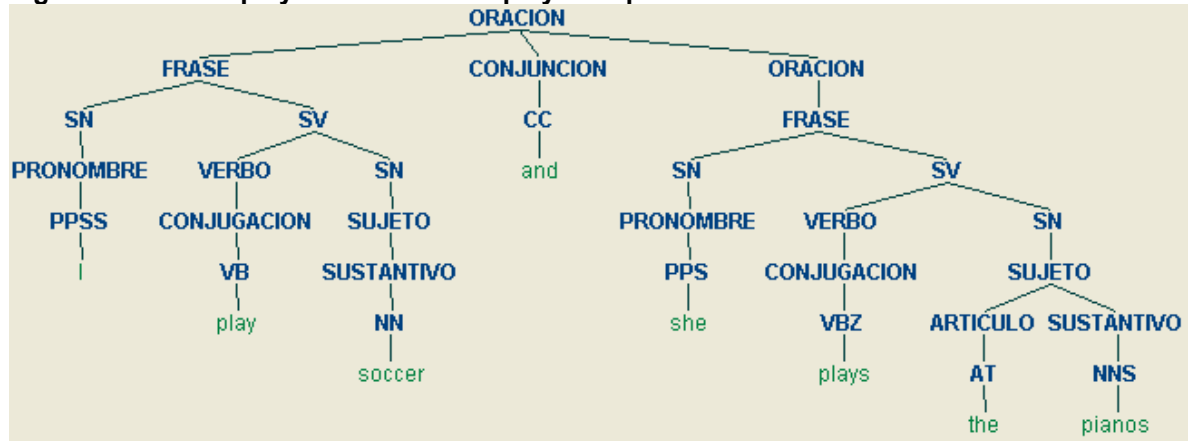
Los significados se pueden deducir sabiendo que *piano* es un instrumento musical y que *soccer* es un juego. De la misma manera en el segundo ejemplo la primera aparición de la palabra *kid* corresponderá a *cabrito* sabiendo que este *bala*¹³⁹. Mientras que la segunda aparición será *niño* ya que este puede *hablar*.

En las siguientes imágenes se muestran los árboles que representan la estructura sintáctica de cada una de las palabras y los cambios que dichos árboles van teniendo a medida que se realiza el proceso, se puede observar que cada uno de los ejemplos esta compuesto por dos frases las cuales tienen la misma estructura gramatical, aún así existen diferentes acepciones para las palabras *play* y *kid* respectivamente.

¹³⁹ Balar, es producir la voz de un carnero, cordero, oveja o cabra.

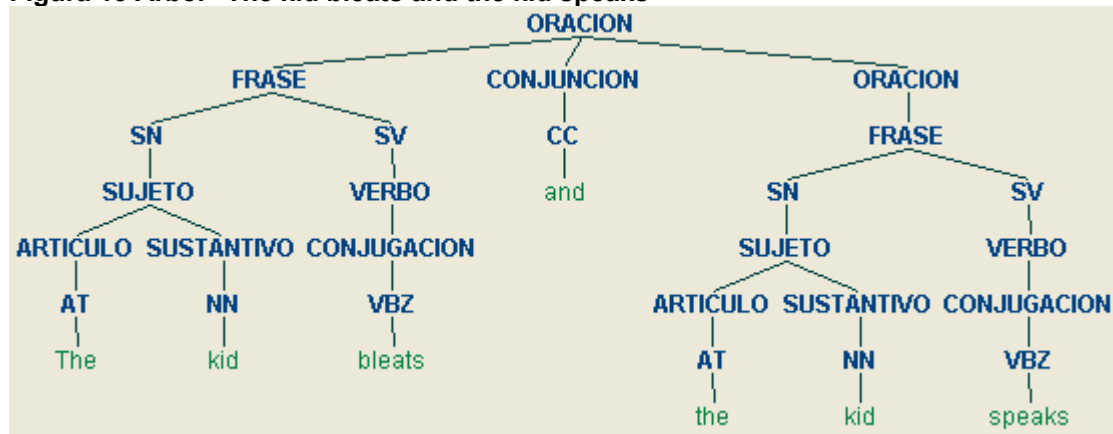
En la Figura 12 se puede observar la estructura retornada por el análisis sintáctico, en el cual se ha identificado *play* como verbo en cada una de las frases. A este nivel no se puede deducir cual de los significados es el que aplica.

Figura 12 Árbol “I play soccer and she plays the pianos”



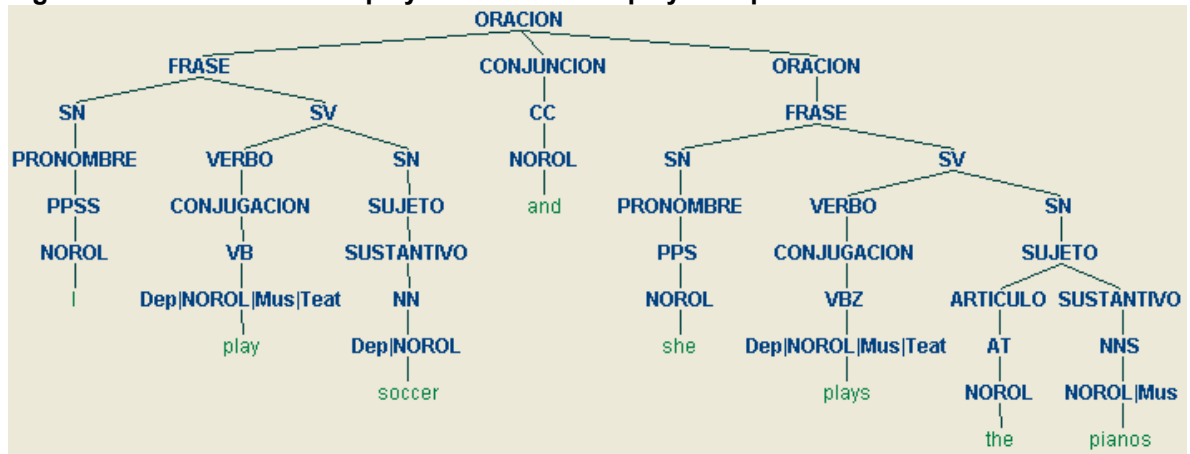
En la Figura 13 se tiene lo mismo que en la imagen anterior, en este ejemplo el análisis sintáctico ha identificado a *kid* como un sustantivo, pero de igual manera aún no se puede ingerir su significado sino se realiza un análisis semántico.

Figura 13 Árbol “The kid bleats and the kid speaks”



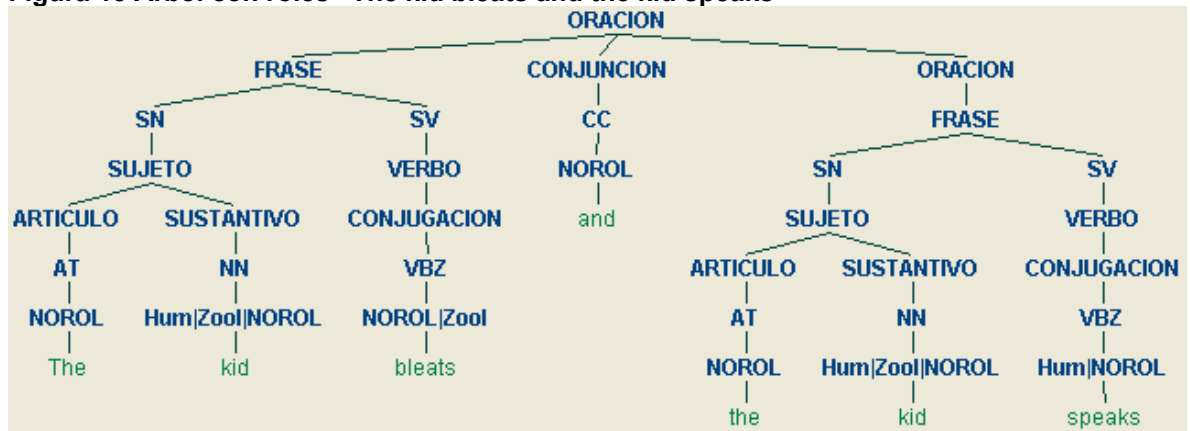
En la Figura 14 se puede observar la misma estructura sintáctica mostrada en la Figura 12 pero en esta se tienen además los roles asociados a cada palabra, este árbol se obtiene después de ejecutar la función `agregar_etiquetas_semanticas`.

Figura 14 Árbol con roles “I play soccer and she plays the pianos”



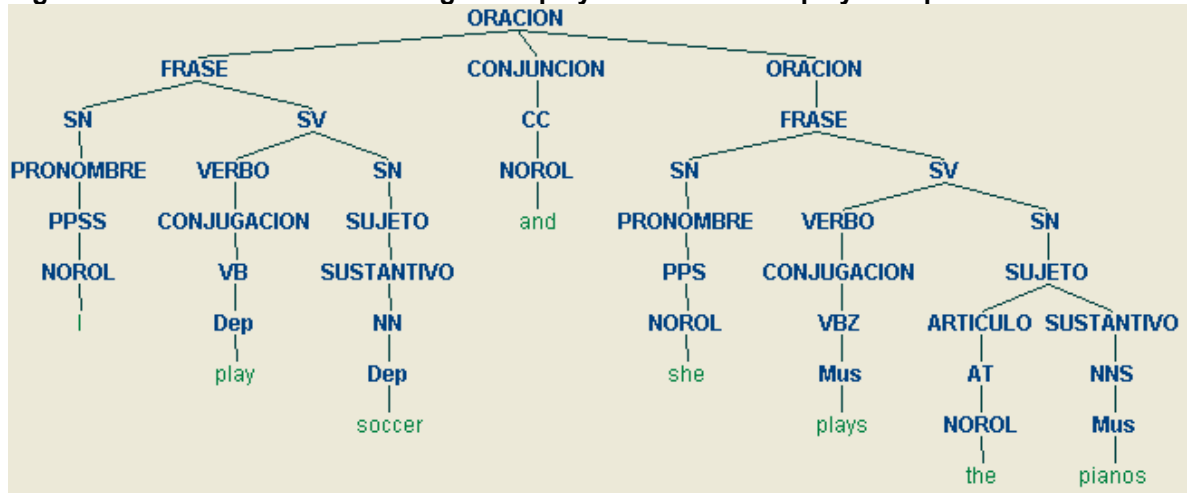
En la Figura 15 se puede observar la misma estructura sintáctica mostrada en la Figura 13 pero en esta se tienen además los roles asociados a cada palabra, este árbol se obtiene después de ejecutar la función `agregar_etiquetas_semanticas`.

Figura 15 Árbol con roles “The kid bleats and the kid speaks”



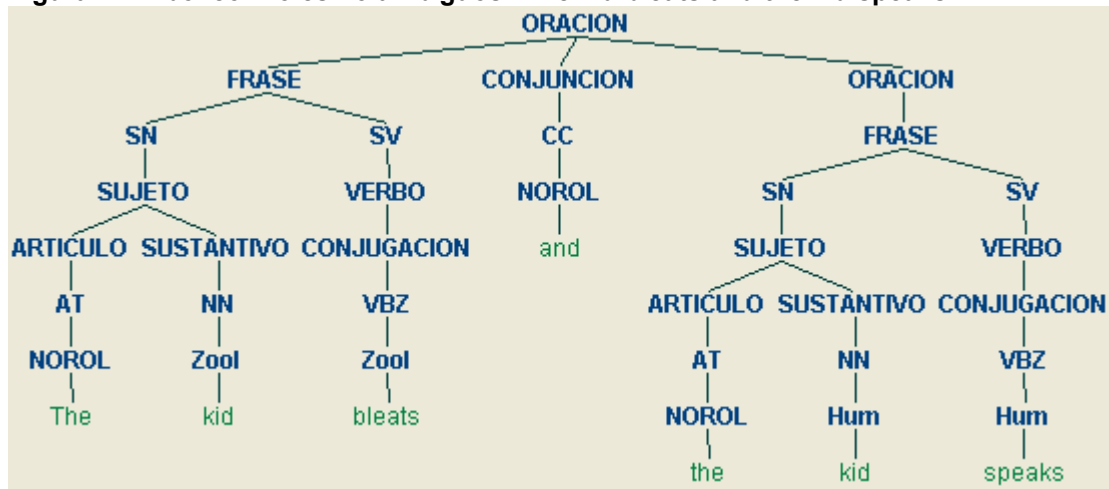
En la Figura 16 se muestra el árbol de la Figura 14 después de ejecutar la función `eliminar_ambigüedades_semanticas`. En este punto se ha dejado como rol aquel que se relaciona con los roles de las demás palabras que componen la frase.

Figura 16 Árbol con roles no ambiguos "I play soccer and she plays the pianos"



Igualmente la Figura 17 es el mismo árbol de la Figura 15 después de ejecutar la función `eliminar_ambigüedades_semanticas`.

Figura 17 Árbol con roles no ambiguos "The kid bleats and the kid speaks"



Como se puede observar, después de ejecutarse el análisis semántico se eliminan problemas que no se habían contemplado en el análisis sintáctico. Para los ejemplos anteriores las categorías y roles han sido asociadas correctamente. Ya en la síntesis se podrá reemplazar de acuerdo a las entradas de cada palabra, etiqueta y rol que se encuentran en el diccionario. De esta manera se entrega a la síntesis una frase sin ambigüedades de orden semántico. La cual puede ser reemplazada de una manera sencilla en el lenguaje destino.

En el presente capítulo se ha realizado una breve introducción al análisis semántico y los problemas asociados a éste. Se trataron los diferentes tipos de conocimientos (semántico, pragmático y del mundo real) que se deben tener en cuenta en el proceso de análisis. Posteriormente se realizó la explicación del prototipo de analizador semántico implementado y se mostró su funcionalidad a través de dos ejemplos en los cuales existía ambigüedad que sólo podía ser resuelta a través de un proceso de análisis semántico.

En capítulo siguiente se tratará la última etapa en el proceso de traducción automática, la síntesis al lenguaje destino que consiste en trasladar la estructura sintáctica y semántica que se tiene de la frase al lenguaje destino. Para realizar este proceso se debe contar con herramientas como un conjugador de verbos y un generador de plurales los cuales serán explicados de una manera general.

9. SÍNTESIS AL LENGUAJE DESTINO

9.1. INTRODUCCIÓN

Como se vio en el capítulo anterior, el análisis semántico, brinda la información final, acerca de lo que se quiere expresar en el lenguaje origen, elimina las ambigüedades basándose en los roles y en el conocimiento semántico como tal. El proceso de traducción se debe finalizar en una etapa de síntesis, tema a tratar en el presente capítulo.

La síntesis consiste en trasladar la estructura sintáctica y semántica que se tiene al lenguaje destino, para ello es necesario un conjugador de verbos, un generador de plurales. Se debe contar además con herramientas que permitan identificar la persona y el número de un sintagma para la selección correcta de la persona de la conjugación, además de la transferencia adecuada para adjetivos, teniendo en cuenta el número y persona. Sumado a esto, se debe tener en cuenta el orden correcto de las palabras en el lenguaje destino.

9.2. ASPECTOS TEÓRICOS

9.2.1. El verbo

El verbo es una categoría gramatical que funciona como núcleo del predicado y suele indicar acción (traer, leer, etc.), proceso (pensar, creer, etc.) o estado (existir, vivir, permanecer, ser, etc.). En español constituye la clase de palabra más variable.

9.2.1.1. La conjugación o flexión verbal

Este es uno de los procesos fundamentales de la síntesis, consiste en adaptar el verbo para los diferentes tiempos verbales. Se llaman tiempos al conjunto de formas verbales que presentan la acción de la misma manera y corresponden a un mismo tiempo, ya sea presente, pasado o futuro. Cada tiempo verbal consta de seis formas que varían en número (singular o plural) y persona. Tanto en el singular como en el plural se tienen 3 personas, para el singular se tiene: *yo, tú, él* y para el plural se tienen: *nosotros, vosotros, ellos*.

Los verbos pueden ser regulares o irregulares, un verbo es regular cuando su conjugación sigue fielmente la de los verbos modelo e irregular cuando no se sigue este modelo.

9.2.1.2. Conjugación de verbos regulares

Se llama conjugación al conjunto de todas las formas de un verbo que resultan de la combinación de un lexema verbal con todas las desinencias verbales posibles. Para el caso del español, que es la lengua destino escogida para analizar, las desinencias de los verbos se distribuyen en tres conjugaciones:

- Primera conjugación: Formada por todos los verbos terminados en *ar*. Por ejemplo: *adivinar, cocinar, amar*.

Conjugación en presente simple para el verbo *adivinar*:

Yo adivino
Tú adivinas
Él adivina
Nosotros adivinamos
Vosotros adivináis
Ellos adivinan

Conjugación para el verbo *cocinar*:

Yo cocino
Tú cocinas
Él cocina
Nosotros cocinamos
Vosotros cocináis
Ellos cocinan

Obsérvese que para cada uno de los tiempos se tiene una particularidad, se suprime la terminación *ar* y se agrega una misma terminación cuyo patrón es igual para todos los pronombres. Para el pronombre *yo* se agrega *o*, para *tú* se agrega *as*, para *él* se agrega *a*, para *nosotros* se agrega *amos*, para *vosotros* se agrega *áis* y para *ellos* se agrega *an*.

- Segunda conjugación: Formada por todos los verbos terminados en *er*. Por ejemplo: *coser, beber, meter*.

Conjugación para el verbo *coser*:

Yo coso
Tú coses
Él cose
Nosotros cosemos
Vosotros coséis
Ellos cosen

Conjugación para el verbo *beber*:

Yo bebo
Tú bebes
Él bebe
Nosotros bebemos
Vosotros bebéis
Ellos beben

Nótese que para cada uno de los tiempos también se tiene una particularidad, se suprime la terminación *er* y se agrega la misma terminación. Para *yo* se agrega *o*, para *tú* se agrega *es*, para *él* se agrega *e*, para *nosotros* se agrega *emos*, para *vosotros* se agrega *éis* y para *ellos* se agrega *en*.

- Tercera conjugación: Pertenecen a ella todos los verbos terminados en *ir*. Por ejemplo: *morir*, *admitir*, *sentir*.

Conjugación para el verbo *morir*:

Yo muero
Tú mueres
Él muere
Nosotros morimos
Vosotros morís
Ellos mueren

Conjugación para el verbo *admitir*:

Yo admito
Tú admites
Él admite
Nosotros admitimos
Vosotros admitís
Ellos admiten

Obsérvese que para cada uno de los tiempos también se tiene una particularidad, se suprime la terminación *ir* y se agrega la misma terminación. Para *yo* se agrega *o*, para *tú* se agrega *es*, para *él* se agrega *e*, para *nosotros* se agrega *imos*, para *vosotros* se agrega *ís* y para *ellos* se agrega *en*.

Todos los verbos regulares se conjugan de alguna de las tres formas antes mencionadas, los verbos que no sigan este modelo se denominan irregulares.

9.2.1.3. Conjugación de verbos irregulares

Los verbos irregulares tienen conjugaciones propias y diferentes en la mayoría de los casos. Para realizar síntesis teniendo en cuenta estos verbos se debe tener un

soporte adicional que permita asignar la conjugación apropiada para el tiempo que se este analizando. Por ejemplo: *ser, dar, estar*.

Conjugación del verbo irregular *estar*:

Yo estoy
Tú estas
Él está
Nosotros estamos
Vosotros estáis
Ellos están

Nótese que este verbo aunque termina en *ar* no cumple con la primera conjugación explicada anteriormente para verbos regulares.

La teoría acerca de los verbos es extensa, pero en este capítulo no se abordará debido a que no es necesaria para la implementación del prototipo.

9.2.2. La formación de plurales

La formación de los plurales a simple vista consiste en agregar una letra *s* al final de la palabra. El tratamiento que se debe dar varía un poco dependiendo de la terminación de la palabra, pero en general el proceso es sencillo.

9.2.2.1. Palabras terminadas en vocal

La formación del plural para palabras terminadas en vocal se forma agregando una *s* al final. Por ejemplo:

cosa: cosas
verde: verdes
vaso: vasos

9.2.2.2. Palabras terminadas en letra *i* con tilde o consonante.

Este tipo de palabras se pluralizan agregando *es* al final. Por ejemplo:

papel: papeles
tenedor: tenedores
marroquí: marroquíes
colibrí: colibríes

Para este caso se tienen dos excepciones.

- Si la consonante en la que termina es una *z*, se reemplaza la *z* por *c* y se agrega *es*. Por ejemplo:

pez: peces

avestruz: avestruces

- Si la palabra termina en una vocal con tilde y una consonante, se elimina la tilde de la vocal y se agrega *es*.

sillón: sillones

composición: composiciones

9.2.2.3. Palabras invariables

Los sustantivos de dos sílabas o más, acabados en *s* o *x* y no agudos no tienen forma plural. Por ejemplo:

lunes, crisis, virus, análisis, énfasis, tórax.

9.2.3. Cambio de género

Dado que en la síntesis debe detectarse correctamente la persona y número que son sujeto de una frase, es necesario que se haga una correspondencia de palabras de género masculino a género femenino cuando así se requiera, es decir, si una persona de género femenino (en el caso del pronombre *she* por ejemplo) actúa sobre un objeto (sustantivo) o tiene un calificativo (adjetivo), las palabras con dichas categorías se deben cambiar a género femenino.

Aunque las reglas son las mismas sin importar la categoría de la palabra, el prototipo implementado solo hace cambio de género para los adjetivos, siguiendo tres reglas para ello.

9.2.3.1. Palabras agudas con tilde

A las palabras agudas que terminan en una vocal tildada seguida de uno de los caracteres *n* o *s*, el cambio de género se hace reemplazando la vocal tildada por una vocal sin acentuación y agregando el carácter *a* al final de la palabra. En la tabla Tabla 3 se pueden ver algunos ejemplos de cambio de género para estos adjetivos de este tipo.

Tabla 3 Cambio de género para adjetivos, palabras agudas

Adjetivo masculino	Adjetivo femenino
Alemán	Alemana
Abusón	Abusona
Burgués	Burguesa
Inglés	Inglesa

9.2.3.2. Terminación en o

Para el caso de las palabras terminadas en *o*, la versión en género femenino se obtiene simplemente reemplazando la *o* por una *a*. Por ejemplo, para el adjetivo masculino *celoso* corresponde el adjetivo femenino *celosa*.

9.2.4. Terminación en or

Al igual que en el caso anterior, la versión en género femenino de las palabras terminadas en *or* se obtiene agregando una *a* al final. Por ejemplo, el adjetivo femenino de la palabra *traductor* es *traductora*.

Si un adjetivo no cumple con una de las tres reglas que se han mencionado, quiere decir que al cambiar de género se obtiene la misma palabra, esto sucede por ejemplo en el caso de la palabra *vulnerabilidad*.

9.3. MODELAMIENTO DE LA SÍNTESIS

9.3.1. Conjugación de Verbos

En este punto es necesario realizar un módulo encargado de la conjugación de los verbos en los diferentes tiempos para las diferentes personas. De tal forma que:

$V_c = \text{Conjugar}(V_b, Su, T)$ dé como resultado el verbo V_c , el cual es el resultado de conjugar el verbo V_b en el tiempo T para la persona Su .

Para la gran mayoría de lenguajes la conjugación se realiza a partir de reglas gramaticales que por lo general son regulares. Aún así hay excepciones en algunos verbos tal como el caso del verbo *to be* en inglés y el verbo *ir* en español en donde la conjugación debe ser tratada como una irregularidad ya que no hay una regla específica que se aplique a este verbo como a otros.

9.3.2. Conformación de plurales

La conformación de plurales, tal como la conjugación de verbos, varía de acuerdo al idioma. En general, un módulo $\text{Plural}(W)$ cumple la siguiente condición:

$W' = \text{Plural}(W)$ Donde W es la palabra a aplicar el plural y W' es el plural de W .

9.3.3. Cambio de género

Este módulo se aplica para idiomas que manejan género como el español. De tal forma que es necesario modificar el género para ciertas palabras que dependen del género de otras (generalmente adjetivos) tal como en la frase *las niñas bonitas* no sería gramaticalmente correcto escribir *las niñas bonitos*. Un módulo $\text{Genero}(W)$ cumple la siguiente condición:

$W' = \text{Genero}(W)$ Donde W es la palabra a la que se quiere cambiar el género (normalmente de masculino a femenino) y W' es la palabra con diferente género de W .

9.3.4. Conjunción de nombres

Un problema complejo y que se presenta en cualquier idioma es la combinación de varios pronombres unidos por una conjunción y realizando una misma acción, es necesario realizar un módulo que a partir de la conjunción de múltiples personas se pueda seleccionar la persona y el número adecuado.

Por ejemplo, en la frase en español *tu y yo estamos en el parque* es necesario identificar que los pronombres *tu* y *yo* conforman la persona nosotros para de esta manera no conjugar el verbo incorrectamente como *tu y yo estas en el parque* o *tu y yo estoy en el parque*.

Debe de realizarse un módulo que permita realizar la conjunción entre dos personas de la siguiente manera:

```
Persona_r, numero_r = persona_numero_conjuncion(persona_1, numero_1,
persona_2, numero_2)
```

Tal que persona_r , numero_r es el resultado de realizar la conjunción de la persona_1 , numero_1 con la persona_2 , numero_2 . Donde persona_i , numero_i son la persona y número (plural o singular) de la persona. El número es necesario manejarlo para identificar conjunciones como *la niña y ella*, que da como resultado *ella* en plural.

9.4. IMPLEMENTACIÓN DEL PROTOTIPO

El prototipo implementado contempla el proceso de síntesis, teniendo en cuenta los aspectos teóricos abordados anteriormente. El prototipo cuenta con una clase principal llamada `Sintesis` la cual se apoya en otra clase llamada `Gramatica` encargada de todo el manejo gramatical del idioma destino que para este caso será el español, para el desarrollo de éste se definieron un conjunto de funciones que serán explicadas a continuación.

9.4.1. Plurales

La función `get_plural` cambia una palabra de singular a plural, recibe como parámetro la palabra a generar el plural, esta función hace parte de la clase `Gramatica` y tiene el siguiente código fuente:

```
def get_plural(self, palabra):
    diccionarioVocalesTilde = KeyInsensitiveDict({'á': 'a', 'é': 'e',
                                                  'í': 'i', 'ó': 'o', 'ú': 'u'})

    patron = re.compile('[aeiou][s|x]$', re.I)
    if (len(patron.findall(palabra)) > 0):
        palabraPlural = palabra
    else:
        patron = re.compile('[áéíóú][n|s]$', re.I + re.UNICODE)
        if (len(patron.findall(palabra)) > 0):
            palabraPlural = palabra[:-2] +
                               diccionarioVocalesTilde[palabra[-2]] +
                               palabra[-1] + 'es'
        else:
            patron = re.compile('z$', re.I)
            if (len(patron.findall(palabra)) > 0):
                palabraPlural = palabra[:-1] + 'ces'
            else:
                patron = re.compile('[ibcdfghjklmnñpqrstvwxyz]$', re.I +
                                     re.UNICODE)
                if (len(patron.findall(palabra)) > 0):
                    palabraPlural = palabra + 'es'
                else:
                    palabraPlural = palabra + 's'
    return palabraPlural
```

Se realiza la definición de la función y se crea un arreglo asociativo que contiene la correspondencia entre las vocales tildadas y no tildadas, para el posterior reemplazo en las reglas.

```
def get_plural(self, palabra):
    diccionarioVocalesTilde = KeyInsensitiveDict({'á': 'a', 'é': 'e',
                                                  'í': 'i', 'ó': 'o', 'ú': 'u'})
```

Las palabras de dos sílabas o más, acabados en 's' o 'x' y no agudas no tienen forma plural. Por ejemplo: *lunes, crisis, virus, análisis, énfasis, tórax*.

Con el siguiente patrón se determina si la palabra recibida cumple o no con esta condición. En caso de cumplirse este patrón la palabra en plural es la misma palabra que se ha recibido.

```
patron = re.compile('[aeiou][s|x]$', re.I)
if (len(patron.findall(palabra)) > 0):
    palabraPlural = palabra
```

En caso de no cumplirse la condición anterior se examina si la palabra termina en una vocal con tilde y una consonante (*n* o *s*), se elimina la tilde de la vocal y se agrega 'es'. Por ejemplo: *sillón-> sillones, composición-> composiciones, vienes-> vieneses*. En este caso el patrón usado incluye además de la bandera de insensitividad de mayúsculas y minúsculas, la bandera para manejo de caracteres *UNICODE*, esto se requiere para que se reconozcan las vocales tildadas en la expresión regular.

```
else:
    patron = re.compile('[áéíóú][n|s]$', re.I + re.UNICODE)
    if (len(patron.findall(palabra)) > 0):
        palabraPlural = palabra[:-2] +
            diccionarioVocalesTilde[palabra[-2]] +
            palabra[-1] + 'es'
```

Si la palabra termina en 'z', se reemplaza la 'z' por 'c' y se agrega 'es'. Por ejemplo: *pez -> peces, avestruz -> avestruces*.

```
else:
    patron = re.compile('z$', re.I)
    if (len(patron.findall(palabra)) > 0):
        palabraPlural = palabra[:-1] + 'ces'
```

Las palabras terminadas en letra 'i' con tilde o consonante se pluralizan agregando 'es' al final. Por ejemplo: *papel-> papeles, tenedor-> tenedores, marroquí-> marroquíes, colibrí-> colibríes*.

```
else:
    patron = re.compile('[íbcdfghjklmnñpqrstvwxyz]$', re.I +
        re.UNICODE)
    if (len(patron.findall(palabra)) > 0):
        palabraPlural = palabra + 'es'
```

La formación del plural para palabras terminadas en vocal se forma agregando una 's' al final. Por ejemplo: *cosa*-> *cosas*, *mamá*-> *mamás*, *vaso*-> *vasos*.

```
else:
    palabraPlural = palabra + 's'
```

Finalmente se retorna el plural.

```
return palabraPlural
```

9.4.2. Géneros

Este módulo implementa la correspondencia del género femenino de una palabra que se encuentra en género masculino. Aunque la implementación actual solo trabaja con la etiqueta adjetivo, se puede generalizar a otras etiquetas como sustantivos, esta función hace parte de la clase `Gramatica` y su código se muestra a continuación.

```
def get_adjetivo_femenino(self, adjetivo):
    listaVocales = 'aeiou'
    diccionarioVocalesTilde = KeyInsensitiveDict({'á': 'a', 'é': 'e',
                                                  'í': 'i', 'ó': 'o', 'ú': 'u'})

    adjetivoFemenino = adjetivo
    terminacionAdjetivo = adjetivo[-1]
    patron = re.compile('[n|s]$', re.I)
    if (len(patron.findall(adjetivo)) > 0):
        if (diccionarioVocalesTilde.has_key(adjetivo[-2])):
            adjetivoFemenino = adjetivo[:-2] +
                               diccionarioVocalesTilde[adjetivo[-2]] +
                               terminacionAdjetivo + 'a'
    else:
        patron = re.compile('o$', re.I)
        if (len(patron.findall(adjetivo)) > 0):
            adjetivoFemenino = adjetivo[:-1] + 'a'
        else:
            patron = re.compile('or$', re.I)
            if (len(patron.findall(adjetivo)) > 0):
                adjetivoFemenino = adjetivo + 'a'
            else:
                adjetivoFemenino = adjetivo
    return adjetivoFemenino
```

Se define la función, una cadena para identificar las vocales y un diccionario insensitivo para las vocales tildadas, además inicializa el adjetivo femenino con el adjetivo masculino recibido como parámetro e inicializa la variable `terminacionAdjetivo` con el último carácter de la palabra.


```
def get_adjetivo_femenino(self, adjetivo):
    listaVocales = 'aeiou'
    diccionarioVocalesTilde = KeyInsensitiveDict({'á': 'a', 'é': 'e',
                                                'í': 'i', 'ó': 'o', 'ú': 'u'})

    adjetivoFemenino = adjetivo
    terminacionAdjetivo = adjetivo[-1]
```

Cuando el adjetivo termina en *n* o *s* y le precede una vocal tildada, se reemplaza dicha vocal por su correspondencia sin acentuación y se agrega *a* a la cadena. Cuando la palabra termina en *n* o *s* y no le precede una vocal con acentuación, se retorna el mismo adjetivo recibido

```
patron = re.compile('[n|s]$', re.I)
if (len(patron.findall(adjetivo)) > 0):
    if (diccionarioVocalesTilde.has_key(adjetivo[-2])):
        adjetivoFemenino = adjetivo[:-2] +
            diccionarioVocalesTilde[adjetivo[-2]] +
            terminacionAdjetivo + 'a'
```

Si en el caso anterior no hubo coincidencia, se evalúa entonces si la terminación del adjetivo es *o*, en este caso se reemplaza el carácter *o* por *a*.

```
else:
    patron = re.compile('o$', re.I)
    if (len(patron.findall(adjetivo)) > 0):
        adjetivoFemenino = adjetivo[:-1] + 'a'
```

Si en el caso anterior no hubo coincidencia, se evalúa entonces si la terminación del adjetivo es *or*, en este caso se añade el carácter *a* al final.

```
else:
    patron = re.compile('or$', re.I)
    if (len(patron.findall(adjetivo)) > 0):
        adjetivoFemenino = adjetivo + 'a'
```

Si el adjetivo no cumple con ninguno de los patrones, entonces en género femenino es igual al género masculino.

```
else:
    adjetivoFemenino = adjetivo
    return adjetivoFemenino
```

9.4.3. Conjugador

Este módulo se encarga de conjugar un verbo en español para un pronombre y tiempo dados. Aunque en la implementación actual sólo se contemplan las conjugaciones en presente simple y gerundio, el módulo es escalable de tal manera que se pueda conjugar un verbo para todos los tiempos.

Para la realización de este módulo fue necesario realizar una tarea de etiquetamiento de alrededor de 9000 verbos del idioma español con el objetivo de identificar los verbos irregulares y crear reglas de conjugación adecuadas para cada uno de los grupos de etiquetas definidos.

La función se llama `Conjugador` y se encuentra en la clase `Gramatica`.

A continuación se muestra el código principal del conjugador que selecciona la función que corresponda a cada uno de los tiempos a conjugar. Si se envía como parámetro un tiempo no definido, se retorna la misma cadena correspondiente al verbo que se recibió.

```
def conjugar(self, sujeto, verbo, tiempo):
    if (tiempo == self.PRESENTESIMPLE):
        return self.conjugar_presente_simple(sujeto, verbo)
    else (tiempo == self.GERUNDIO):
        return self.conjugar_gerundio(verbo)
    else:
        return verbo
```

Para el caso de la conjugación en presente simple se usa la función `conjugar_presente_simple`. Esta función primero determina si el verbo es irregular de difícil formalización mediante reglas, de ser así, retorna la conjugación correspondiente, si el verbo es irregular pero tiene asociado una regla de conjugación, entonces se aplica dicha regla y se retorna la conjugación correspondiente, en caso contrario se conjuga el verbo mediante las reglas de conjugación de verbos irregulares del español.

```
def conjugar_presente_simple(self, sujeto, verbo):
    diccionarioPronombres = KeyInsensitiveDict({'yo': '', 'tú': '',
                                                'eso': '', 'nosotros': '',
                                                'ellos': ''})

    verbosIrregulares = KeyInsensitiveDict({'ser': '', 'estar': '',
                                             'haber': '', 'ir': ''})

    if (verbosIrregulares.has_key(verbo)):
        return self._conjugar_irregular(sujeto, verbo,
                                         self.PRESENTESIMPLE)

    diccionarioIrregularidades = self._cargar_irregularidades(
        environ['ARCHIVOS_TRADUCTOR'] + sep +
```

```

                                'Irregularidades.txt')
etiquetaVerbo = self._buscar_etiqueta(verbo,
                                       environ['ARCHIVOS_TRADUCTOR'] + sep + 'Verbos.txt')
pronombre = sujeto
if (not diccionarioPronombres.has_key(sujeto)):
    pronombre = 'eso'
if (diccionarioIrregularidades.has_key(pronombre)):
    if (diccionarioIrregularidades[pronombre].has_key(verbo[-2:])):
        if (diccionarioIrregularidades[pronombre][verbo[-2:]].\
            has_key(etiquetaVerbo)):
            listaReemplazos = diccionarioIrregularidades[pronombre]\
                               [verbo[-2:]][etiquetaVerbo]
            tuplaTerminaciones = listaReemplazos[0]
            tuplaIntermedio = listaReemplazos[1]
            patron = re.compile(tuplaTerminaciones[0]+'$', re.I)
            if (len(patron.findall(verbo)) > 0):
                patron = re.compile('(' + tuplaIntermedio[0] +
                                     '(!r$))', re.I + re.UNICODE)
                verboSeparado = patron.split(verbo)
                conjugacion = verbo
                if (len(verboSeparado) > 1):
                    verboSeparado[-2] = tuplaIntermedio[1]
                    conjugacion = ''.join(verboSeparado)
                conjugacion = self._cambiar_terminacion(conjugacion,
                                                         tuplaTerminaciones[0],
                                                         tuplaTerminaciones[1])

            return conjugacion
return(self._conjugar_general(sujeto, verbo, self.PRESENTESIMPLE))

```

Se define la lista de pronombres del español. El pronombre por defecto es 'eso'. Al no incluir 'él' y 'ella' en el diccionario de pronombres, asumirán entonces el pronombre por defecto. Nótese que no se genera una conjugación para el pronombre vosotros ya que no tiene una correspondencia con algún pronombre del inglés.

```

def conjuguar_presente_simple(self, sujeto, verbo):
    diccionarioPronombres = KeyInsensitiveDict({'yo': '', 'tú': '',
                                                'eso': '', 'nosotros': '',
                                                'ellos': ''})

```

Se define el arreglo asociativo de verbos irregulares de difícil formalización, su conjugación se hace directamente.

```

verbosIrregulares = KeyInsensitiveDict({'ser': '', 'estar': '',
                                         'haber': '', 'ir': ''})

```

Primero se evalúa si el verbo es irregular de difícil formalización, caso en el cual se retorna su conjugación fija, entregada por la función `conjuguar_irregular`.

```

if (verbosIrregulares.has_key(verbo)):
    return self._conjugar_irregular(sujeto, verbo,
                                     self.PRESENTESIMPLE)

```

Carga el archivo que contiene las reglas para conjugación de verbos irregulares. Es necesario que se tenga definida la variable de entorno ARCHIVOS_TRADUCTOR.

La estructura del archivo *Irregularidades.txt* es:

Pron : Term : Etiq : TermI - TermF / InterI - InterF

Siendo:

Pron: Pronombre personal.

Term: Terminación del verbo.

Etiqu: Etiqueta asignada al verbo en el archivo *Verbos.txt*.

TermI: Terminación que debe reemplazarse.

TermF: Terminación por la que debe reemplazarse.

InterI: Reemplazo intermedio de caracteres que debe cambiar.

InterF: Reemplazo intermedio de caracteres fruto del cambio.

Un ejemplo de una línea del archivo de irregularidades es:

Yo : er : B : er - o / e - ie

Quiere decir la anterior línea que para los verbos terminados en *en*, cuya etiqueta en el archivo de verbos sea *B* y cuando el pronombre personal sea *yo*, se les cambia la terminación *er* por *o* y si internamente tienen una *e*, se cambia por *ie*.

```

diccionarioIrregularidades = self._cargar_irregularidades(
    environ['ARCHIVOS_TRADUCTOR'] + sep +
    'Irregularidades.txt')

```

Se lee la etiqueta que se ha definido para el verbo en el archivo de verbos.

```

etiquetaVerbo = self._buscar_etiqueta(verbo,
    environ['ARCHIVOS_TRADUCTOR'] + sep + 'Verbos.txt')

```

Si se recibe como parámetro un pronombre diferente a los establecidos, se asume entonces como *eso* y se hace la conjugación correspondiente a este último.

```

pronombre = sujeto
if (not diccionarioPronombres.has_key(sujeto)):
    pronombre = 'eso'

```

Se evalúa si para el verbo que se va a conjugar hay una regla que involucre el pronombre, verbo y etiqueta, de ser así, esto quiere decir que el verbo es irregular y que debe conjugarse de acuerdo a las reglas que se han determinado en el archivo de irregularidades.

`listaReemplazos` es una lista que consta de dos tuplas, la primera (terminaciones) indica las terminaciones verbales origen y destino y la segunda (intermedio) indica el reemplazo que debe hacerse en los caracteres intermedios del verbo.

```
if (diccionarioIrregularidades.has_key(pronombre)):
    if (diccionarioIrregularidades[pronombre].has_key(verbo[-2:])):
        if (diccionarioIrregularidades[pronombre][verbo[-2:]].\
            has_key(etiquetaVerbo)):
            listaReemplazos = diccionarioIrregularidades[pronombre]\
                               [verbo[-2:]][etiquetaVerbo]
            tuplaTerminaciones = listaReemplazos[0]
            tuplaIntermedio = listaReemplazos[1]
```

Si la terminación coincide con la del verbo, se hace el reemplazo intermedio primero y luego el reemplazo al final. La expresión regular usada identifica si la terminación del verbo coincide con la de la regla de reemplazo en cuanto a la terminación, de ser así, se aplican las reglas de reemplazo terminal e intermedio.

```
patron = re.compile(tuplaTerminaciones[0]+'$', re.I)
if (len(patron.findall(verbo)) > 0):
```

Para el reemplazo intermedio, debe ubicarse la cadena que coincida con el patrón del reemplazo que esté más a la derecha y que no coincida con la terminación del verbo. Por ejemplo, para el verbo *defender* la conjugación para el pronombre *yo* debe dar como resultado *defiendo*. Para esto, la regla de reemplazo en cuanto a la terminación del verbo dice que se debe cambiar *er* por *o* y la regla de reemplazo intermedio dice que se debe cambiar la vocal *e* por *ie*. Este verbo contiene tres apariciones de la vocal *e*, así que debe definirse cuál es la que debe reemplazarse. Para esto, la expresión regular empleada a continuación busca la aparición del patrón de reemplazo que se encuentre más a la derecha y que no coincida con el fin del verbo. Así, se logra identificar que la *e* que debe reemplazarse es la que aparece en la sílaba *fén*.

```
patron = re.compile('(' + tuplaIntermedio[0] +
                    '(!r$))', re.I + re.UNICODE)
```

Siguiendo con el ejemplo del verbo *defender*, en este caso `verboSeparado` queda con el valor: `['d', 'e', 'f', 'e', 'nder']`.

La salida de la conjugación en cuanto al reemplazo intermedio será inicialmente la misma entrada en el caso de que para el verbo a conjugar no exista una regla de reemplazo intermedio pues esta es opcional.

```
verboSeparado = patron.split(verbo)
conjugacion = verbo
```

Cuando se encuentra coincidencia del patrón intermedio, la separación del verbo produce por lo menos una lista de dos elementos. El elemento más a la derecha (no siendo el último) es el que debe reemplazarse. El penúltimo elemento de la lista se accesa con el índice -2.

```
if (len(verboSeparado) > 1):
    verboSeparado[-2] = tuplaIntermedio[1]
    conjugacion = ''.join(verboSeparado)
```

Independientemente de si se hizo o no el reemplazo intermedio, el reemplazo del fin de la cadena siempre se hace y para eso se usa la función `_cambiar_terminacion` que simplemente se encarga de reemplazar el final de una cadena por otra subcadena.

```
conjugacion = self._cambiar_terminacion(conjugacion,
                                         tuplaTerminaciones[0],
                                         tuplaTerminaciones[1])
```

Si el verbo se conjugó aplicando reglas de conjugación para verbos irregulares, se retorna el resultado de dicha conjugación, de lo contrario se conjuga de acuerdo a las reglas de reemplazos para verbos regulares.

```
return conjugacion
return(self._conjugar_general(sujeto, verbo, self.PRESENTESIMPLE))
```

Dado que el módulo mismo tiene más implementación que la que aquí se muestra, al igual que en el módulo principal se recomienda revisar el código fuente con su documentación asociada y las API's para tener una idea más completa de la implementación.

9.4.4. Múltiples pronombres

A continuación se presenta una explicación para resolver las combinaciones de varios pronombres unidos por una conjunción y realizando una misma acción, se elaboró un algoritmo que permite seleccionar la persona y el número adecuado.

El algoritmo carga la información requerida de un archivo plano llamado *ConjuncionPron.txt*, ubicado en la carpeta de archivos de texto definida por la variable de entorno *ARCHIVOS_TRADUCTOR*, este archivo contiene todas las posibles combinaciones de las personas del inglés. La correspondencia se muestra en la Tabla 4:

Tabla 4 Correspondencia de persona y número

Persona - Número	Correspondencia
I	1
You	2
He	3
She	4
It	5
We	6
They	7
You (Plural)	8
He (Plural)	9
She (Plural)	10
It (Plural)	11

El archivo *ConjuncionPron.txt* contiene tres elementos separados por guión, el primer y segundo elemento indican las personas que están interviniendo y el tercer elemento indica el resultado que se obtiene de la conjunción.

Una línea del archivo *Conjunciones.txt* sería:
2-3-8

Esta línea indica que la conjunción entre 2 y 3, o sea, entre *You* y *He* da como resultado 8, o sea *You* en plural.

El algoritmo permite identificar con certeza todas las posibles conjunciones entre las 11 personas. Se puede observar como *I*, *We* y *They* no tienen plural, ya que no aplica para estos casos.

El código de la explicación anterior se muestra a continuación.

```
def _persona_numero_conjuncion(self, persona1, numero1,
                               persona2, numero2):
    personasInsensitivas = KeyInsensitiveDict({'i': 'i', 'you': 'you',
                                                'he': 'he', 'she': 'she',
                                                'it': 'it', 'we': 'we',
                                                'they': 'they'})
```

```

personal = personasInsensitivas[personal]
persona2 = personasInsensitivas[persona2]

correspondenciaPersonas = {('i', 'Singular'): '1',
                           ('you', 'Singular'): '2',
                           ('you', 'Plural'): '8',
                           ('he', 'Singular'): '3',
                           ('he', 'Plural'): '9',
                           ('she', 'Singular'): '4',
                           ('she', 'Plural'): '10',
                           ('it', 'Singular'): '5',
                           ('it', 'Plural'): '11',
                           ('we', 'Singular'): '6',
                           ('we', 'Plural'): '6',
                           ('they', 'Singular'): '7',
                           ('they', 'Plural'): '7'}

PN1 = correspondenciaPersonas[(personal, numero1)]
PN2 = correspondenciaPersonas[(persona2, numero2)]
archivo = Archivo(environ['ARCHIVOS_TRADUCTOR'] + sep +
                  'ConjuncionPron.txt', 'r')

archivo.leer()
for i in range(archivo.get_num_lineas()):
    linea = archivo.get_linea()
    componentes = linea.split('-')
    if (componentes[0] == PN1 and componentes[1] == PN2) or\
        (componentes[0] == PN2 and componentes[1] == PN1):
        resultado = componentes[2]
    for llave in correspondenciaPersonas.keys():
        if correspondenciaPersonas[llave] == resultado:
            return llave

```

Se crea un arreglo asociativo insensitivo `personasInsensitivas` que tiene como llaves los diferentes pronombres personales y a cada llave se le asocia el valor del pronombre en minúsculas.

```

def _persona_numero_conjuncion(self, personal, numero1,
                               persona2, numero2):
    personasInsensitivas = KeyInsensitiveDict({'i': 'i', 'you': 'you',
                                                'he': 'he', 'she': 'she',
                                                'it': 'it', 'we': 'we',
                                                'they': 'they'})

```

Tanto `persona1` como `persona2` se cargan con el valor de la persona en minúscula, esto con el fin de que en el arreglo asociativo `correspondenciaPersonas` no haya problemas de sensibilidad a mayúsculas.

```

personal = personasInsensitivas[personal]
persona2 = personasInsensitivas[persona2]

```


Se define un arreglo asociativo que tiene como llaves los diferentes pares ordenados (persona, número) y como valor un número único, esto con el fin de clasificar cada uno de los diferentes pares ordenados que se pueden presentar y luego definir de manera más sencilla el resultado de realizar una conjunción entre cualquiera de ellos

```
correspondenciaPersonas = { ('i', 'Singular'): '1',  
                             ('you', 'Singular'): '2',  
                             ('you', 'Plural'): '8',  
                             ('he', 'Singular'): '3',  
                             ('he', 'Plural'): '9',  
                             ('she', 'Singular'): '4',  
                             ('she', 'Plural'): '10',  
                             ('it', 'Singular'): '5',  
                             ('it', 'Plural'): '11',  
                             ('we', 'Singular'): '6',  
                             ('we', 'Plural'): '6',  
                             ('they', 'Singular'): '7',  
                             ('they', 'Plural'): '7' }
```

Se asigna a las variables PN1 y PN2 el número correspondiente a la personas y números recibidos.

```
PN1 = correspondenciaPersonas[(persona1, numero1)]  
PN2 = correspondenciaPersonas[(persona2, numero2)]
```

Se abre el archivo de correspondencias de conjunción de pronombres *ConjuncionPron.txt*.

```
archivo = Archivo(environ['ARCHIVOS_TRADUCTOR'] + sep +  
                  'ConjuncionPron.txt', 'r')  
archivo.leer()
```

Se examinan las combinaciones de PN1 y PN2 a través del archivo *ConjuncionPron.txt* el cual contiene las diferentes correspondencias que se pueden presentar entre las parejas ordenadas (persona, número).

```
for i in range(archivo.get_num_lineas()):  
    linea = archivo.get_linea()  
    componentes = linea.split('-')  
    if (componentes[0] == PN1 and componentes[1] == PN2) or\  
        (componentes[0] == PN2 and componentes[1] == PN1):  
        resultado = componentes[2]
```

El resultado de realizar la conjunción es un número, este tiene su reciproco en el arreglo asociativo `correspondenciaPersonas`, mediante un recorrido en el

diccionario se identifica el par ordenado (persona, numero) que le corresponde. Al encontrar dicho valor se retorna.

```
for llave in correspondenciaPersonas.keys():
    if correspondenciaPersonas[llave] == resultado:
        return llave
```

9.4.5. Función Principal

La función principal es `sintetizar` y hace parte de la clase `Sintesis` cuyo constructor instancia el análisis semántico, el diccionario y el analizador morfológico.

Para cada uno de los árboles se ejecuta la función `realizar_sintesis_arbol`, la cual como su nombre lo indica realiza la síntesis de un árbol. Este proceso se hace recursivamente a través de cada uno de los subárboles que componen el árbol recibido. Dada la extensión del código fuente no se hará explicación detallada de los bloques que lo constituyen. Se aconseja remitirse al código original y revisar la documentación allí contenida.

```
def sintetizar(self):
    self.arbolesSintesis = []
    for arbolSemantico in self.semantico.get_arboles():
        arbolSintesis = self._realizar_sintesis_arbol(arbolSemantico)
        self.arbolesSintesis.append(arbolSintesis)
    return self.arbolesSintesis
```

A continuación se realiza la explicación del análisis de uno de los nodos que pueden ser recibidos y las diferentes acciones que se toman de acuerdo a los hijos que pueden constituirlos, todo esto de acuerdo a la gramática definida en el archivo *Gramatica.txt* que se encuentra en la carpeta asociada a la variable de entorno *ARCHIVOS_TRADUCTOR*.

Si el nodo del árbol recibido es la etiqueta correspondiente a *FRASE* puede ocurrir que los hijos que lo constituyen sean *SN* o *SN SV*. Para el primer caso, el primer hijo será igual al resultado de realizar la síntesis a este hijo. Para el segundo caso, de igual manera el primer hijo será igual al resultado de realizar la síntesis a este hijo. Posteriormente, se encuentra la persona y número del sintagma nominal para enviarlos al proceso de síntesis del sintagma verbal ya que este último es modificado por el primero.

```
else nodoArbol == 'FRASE':
    '''FRASE -> SN | SN SV'''
    if hijosArbol == ['SN']:
```

```

        arbolSintesis[0] = self._realizar_sintesis_arbol(
            arbolSemantico[0])
    else hijosArbol == ['SN', 'SV']:
        (persona, numero) = self._persona_numero(arbolSemantico[0])
        arbolSintesis[0] = self._realizar_sintesis_arbol(
            arbolSemantico[0])
        arbolSintesis[1] = self._realizar_sintesis_arbol(
            arbolSemantico[1], persona, numero)

```

Como se puede observar la sintaxis del programa permite identificar claramente las acciones que se deben tomar para cada uno de los diferentes nodos con sus respectivos hijos que se pueden encontrar. Estas acciones dependen de la gramática definida en el archivo *Gramatica.txt*, siendo éste un archivo editable se deduce entonces que la gramática puede ser cambiada teniendo en cuenta que la creación de nuevas reglas de producción que incluyan etiquetas no consideradas en el archivo inicial implica cambiar el código fuente.

La implementación del prototipo consiste en analizar las oraciones recursivamente, trata de identificar el sintagma nominal y el sintagma verbal, para así seleccionar la conjugación adecuada de acuerdo a la persona y número contenidos en el sintagma nominal. Esto quiere decir que es necesario identificar con qué persona va a ser conjugado el verbo, información que se encuentra contenida en el sintagma nominal. Al identificar el tipo de persona se procede con el proceso de conjugación ya descrito.

En la Tabla 5 se observan las entradas y salidas para algunos ejemplos en el proceso de síntesis.

Tabla 5 Entrada y salida del proceso de síntesis

Entrada (inglés)	Salida (español)
I play soccer and you play piano	Yo juego fútbol y tu juegas piano
I play soccer and she plays the pianos	Yo juego fútbol y ella toca los pianos
Sebastian works with Tokenization	Sebastian trabaja con Tokenization
I'm working with Tokenization	Yo estoy trabajando con tokenization
Jose Ferney Franco fishes in the river	Jose Ferney Franco pesca en el río
The young boy makes jokes	El niño joven hace bromas
I button the button	Yo abotono el botón
The computer is an electronic device	El computador es un dispositivo electrónico
The kid plays with the pianos and the girl plays with dolls	El niño toca con los pianos y la niña juega con muñecas
The kid bleats and the kid speaks	El cabrito bala y el niño habla
The kid plays piano	El niño toca piano

The kid plays with the girl	El niño juega con la niña
I work tokenization with Sebastian	Yo trabajo tokenization con Sebastian
The kid plays a match with the girl and the kid plays a piano with the girl	El niño juega un partido con la niña y el niño toca un piano con la niña
New York is a city	Nueva York es una ciudad

En el presente capítulo se hizo una introducción teórica y práctica al proceso síntesis al lenguaje destino en TA, el cual consiste en trasladar la estructura sintáctica y semántica que se tiene al lenguaje destino, para ello se deben considerar factores como la conjugación de verbos y la generación de plurales. Se realizó una descripción teórica de este tipo de problemas y la manera de solucionarlos. Igualmente se mostró la funcionalidad a través de un prototipo y se mostraron los resultados obtenidos al realizar la síntesis a ejemplos prácticos. Estos ejemplos fueron probados en traductores gratuitos en Internet obteniendo una traducción equivocada ya que no se contempla la información semántica y en algunos casos incluso tienen problemas con la información sintáctica.

La síntesis es la última etapa en el proceso de traducción automática, aunque algunos autores incluyen una etapa de post edición en la cual se refina el documento obtenido. Puede decirse que el proceso de traducción automática termina en este punto.

En el siguiente capítulo se mostrará un prototipo gráfico que contempla todos los prototipos desarrollados, con opciones de visualización que permiten comprender el funcionamiento de cada prototipo.

10. PROTOTIPO GRÁFICO

Para hacer más amigable la interacción del usuario con los prototipos, se generó una IGU (Interfaz Gráfica de Usuario), la cual sencillamente permite ver el proceso de traducción paso a paso y observar la transformación que va teniendo un texto de entrada a través del proceso de traducción hasta la salida final.

Al ejecutarse la IGU, se observará una ventana como la que se observa en la Figura 18:

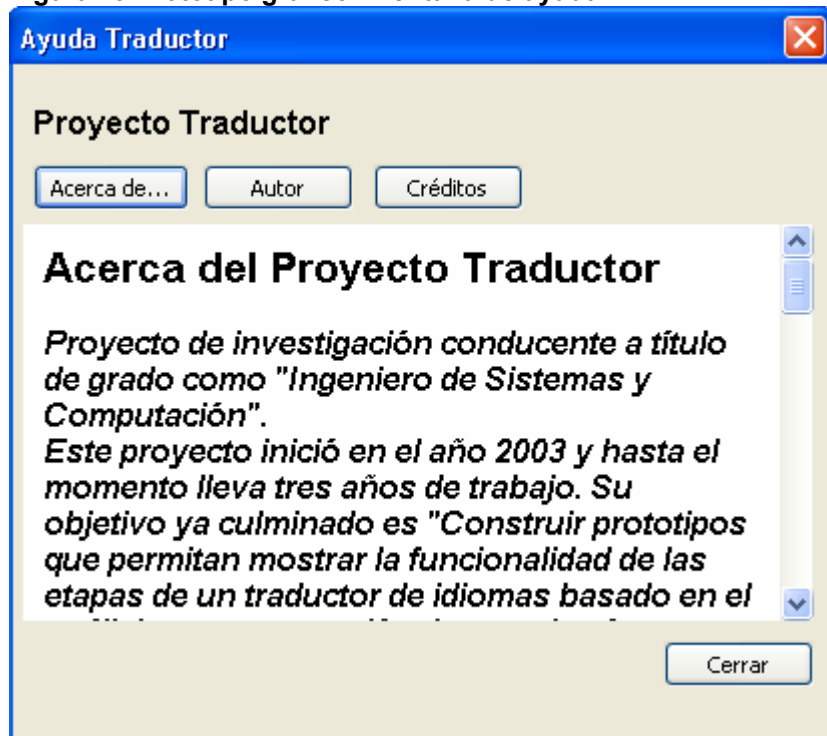
Figura 18 Prototipo gráfico - Ventana Inicial



En esta ventana se podrá elegir una de tres opciones: “VER DEMOSTRACIÓN” para acceder a la demostración de los prototipos, “SALIR” para abandonar la ventana o el botón de ayuda “?” para obtener información general del proyecto.

Si la opción elegida es la ayuda se desplegará una ventana de información acerca del proyecto, los autores y los créditos, como se observa en la Figura 19:

Figura 19 Prototipo gráfico - Ventana de ayuda



Si la opción elegida es "VER DEMOSTRACIÓN" se desplegará la ventana principal, como se observa en la Figura 20:

Figura 20 Prototipo gráfico - Ventana principal



Esta ventana tiene diversas opciones, que se resumen en el proceso mismo de traducción automática, que va desde la entrada en el lenguaje origen hasta la salida en el lenguaje destino, pasando por el preprocesamiento, tokenización, análisis morfológico, análisis sintáctico, análisis semántico y síntesis al lenguaje destino. Cabe anotar que el lenguaje origen para este prototipo de traductor es el inglés y lenguaje destino es el español.

Además de existir botones para las opciones de la entrada, la salida que sería la traducción misma y todo el proceso de traducción, también hay botones de ayuda para cada módulo y un botón llamado “SALIR” para abandonar la ventana.

El primer paso en esta ventana es definir la entrada, que deberá ser una oración en presente simple, o un archivo con una oración. Para ingresar la entrada se debe hacer clic en el botón “ENTRADA”, así se desplegará una ventana como se observa en la Figura 21:

Figura 21 Prototipo gráfico - Entrada de texto

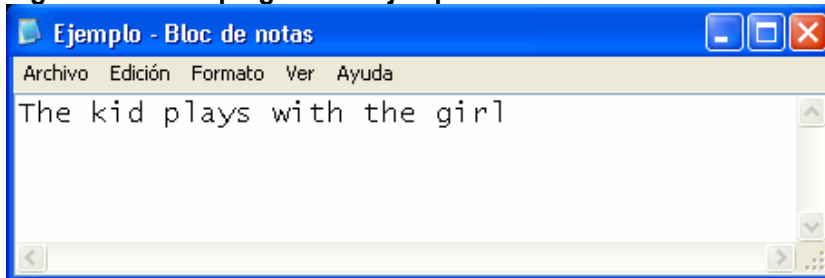
El prototipo gráfico muestra una ventana titulada "Entrada de texto" con un botón de cierre en la esquina superior derecha. El contenido de la ventana está dividido en tres secciones principales:

- Texto a traducir:** Una etiqueta que precede a un campo de texto rectangular. El campo contiene el texto "The kid plays with the girl" y tiene botones de desplazamiento vertical en su extremo derecho.
- Archivo a traducir:** Una etiqueta que precede a un botón etiquetado "Archivo...".
- Quiero Traducir:** Una etiqueta que precede a un grupo de botones de opción. El grupo contiene dos opciones: "Texto" (seleccionada, indicada por un círculo con un punto verde) y "Archivo" (no seleccionada, indicada por un círculo vacío).

En la parte inferior central de la ventana, hay un botón etiquetado "Aceptar".

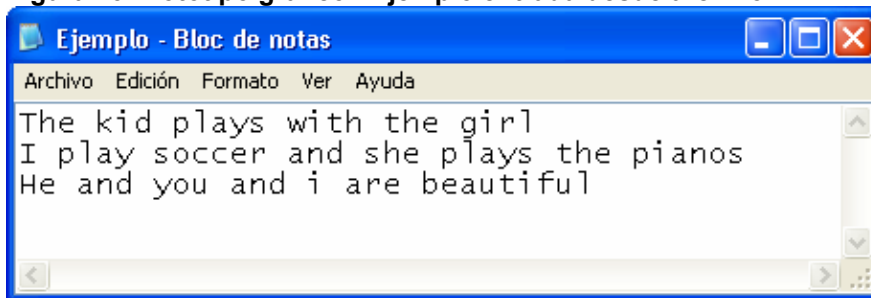
Nótese como aparece una oración por defecto en el campo de texto a traducir: *"The kid plays with the girl"*, la cual puede ser modificada si así se desea, en esta ventana también está la opción de elegir un archivo para traducir, el cual deberá ser necesariamente un archivo con una línea de texto en el caso de requerir observar el funcionamiento de los módulos análisis sintáctico, análisis semántico, síntesis y la salida, ya que estos prototipos soportan solo una oración como entrada, como se observa en la Figura 22:

Figura 22 Prototipo gráfico - Ejemplo entrada desde archivo I



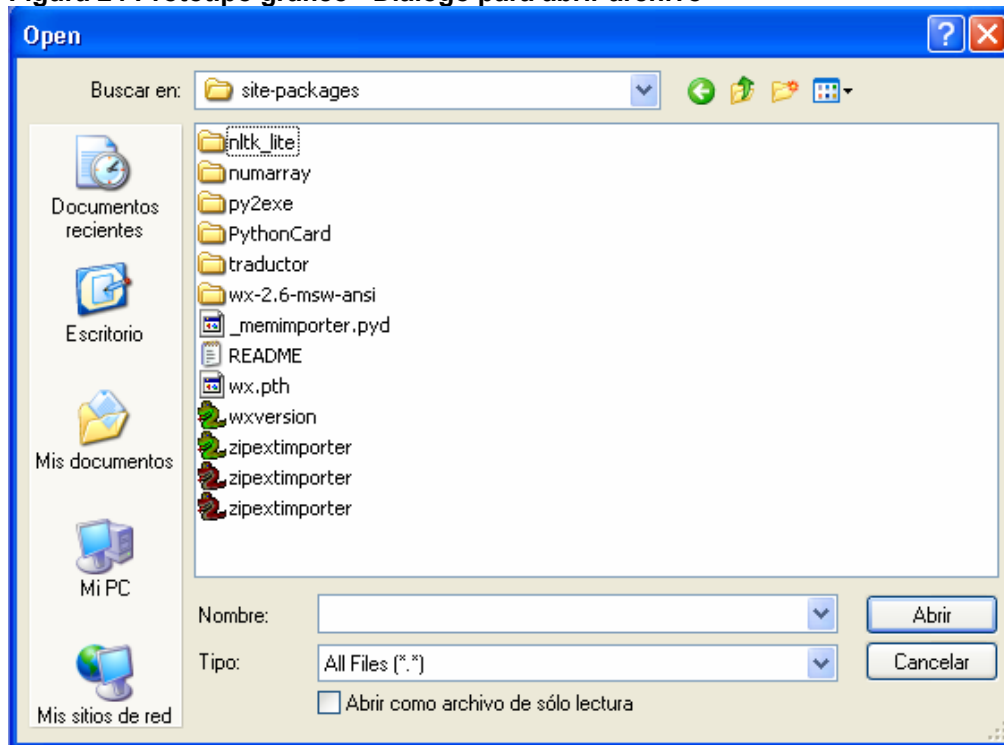
Para el caso de requerir observar el funcionamiento de los prototipos de preprocesamiento, tokenización y análisis morfológico el texto puede ser tan complejo como se desee. Pero, para el análisis morfológico las oraciones que estén compuestas de tokens diferentes a palabras serán etiquetados como nombres propios (esto se debe al alcance definido en el prototipo). Así, una entrada podría ser como se observa en la Figura 23:

Figura 23 Prototipo gráfico - Ejemplo entrada desde archivo II



El archivo podrá ser escogido en una ventana de diálogo como se observa en la Figura 24, este no necesariamente deberá tener la extensión .txt, el único requisito es que sea un archivo de texto plano, pues de lo contrario se producirán errores en todas las etapas:

Figura 24 Prototipo gráfico - Diálogo para abrir archivo



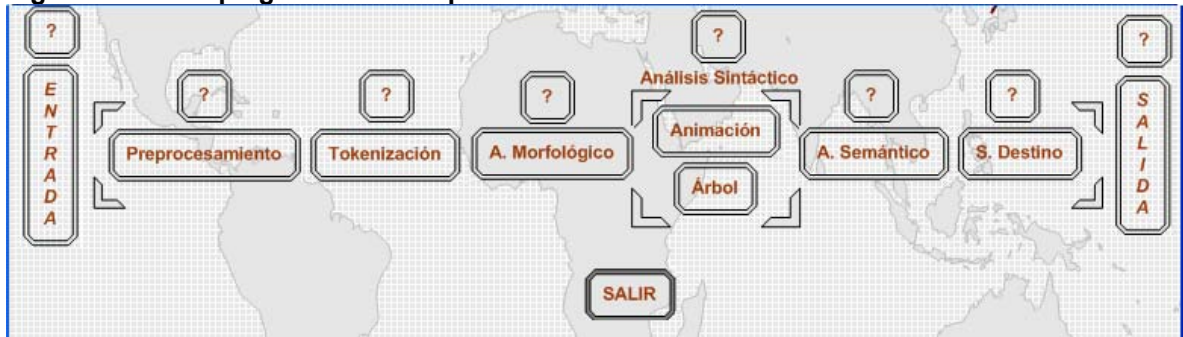
Seguidamente está la opción de escoger que acción se desea realizar, ya sea traducir el texto o traducir el archivo, y un botón para aceptar y pasar a la ventana principal, como se observa en la Figura 25:

Figura 25 Prototipo gráfico - Selección fuente de traducción



Los siguientes botones de la ventana principal correspondientes al proceso de traducción, y a la salida o traducción se pueden pulsar indistintamente y el resultado de hacer clic en cada uno de ellos será el acumulado del proceso de traducción hasta ese punto, el menú del proceso de traducción se observa en la Figura 26:

Figura 26 Prototipo gráfico - Menú proceso de traducción



Si se oprime el botón de preprocesamiento en la ventana principal se desplegará una ventana con el resultado de dicho proceso, para este prototipo la entrada puede ser una o varias oraciones, que pueden ir separadas por signos de puntuación, como se observa en la Figura 27:

Figura 27 Prototipo gráfico - Salida preprocesamiento

Si se oprime el botón de tokenización en la ventana principal se desplegará una ventana con el resultado de dicho proceso como se observa en la Figura 28, el resultado es un listado de los tokens del texto de entrada, es decir, cada una de las piezas de texto que lo conforman:

Figura 28 Prototipo gráfico - Salida tokenización

Salida Tokenización

Texto de entrada

Brad Smith, Microsoft's general
counsel , called the decision to license
parts.

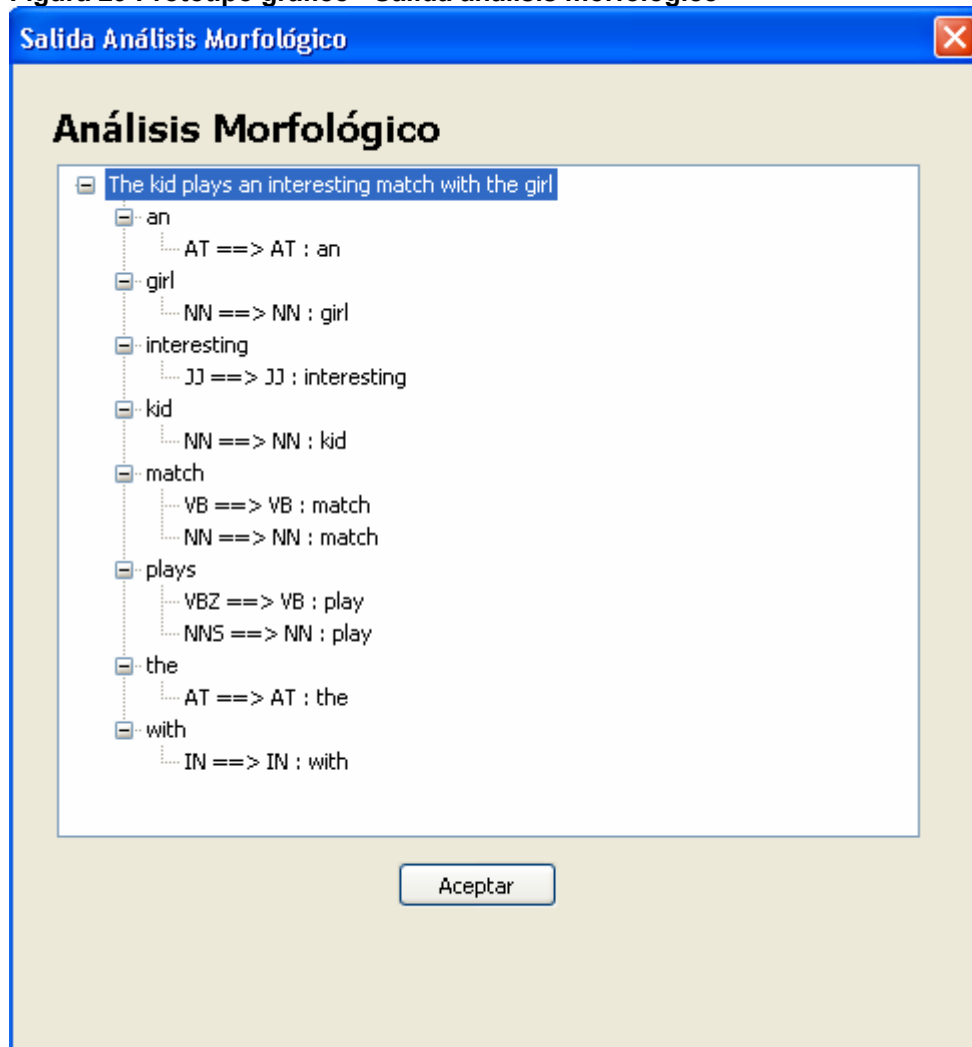
Lista de tokens

Brad
Smith
,
Microsoft's
general

Aceptar

Si se oprime el botón de análisis morfológico en la ventana principal se muestra la salida de este proceso como se observa en la Figura 29, que corresponde a un árbol cuyos hijos son los tokens y cuyos nietos son la clasificación morfológica de los hijos:

Figura 29 Prototipo gráfico - Salida análisis morfológico



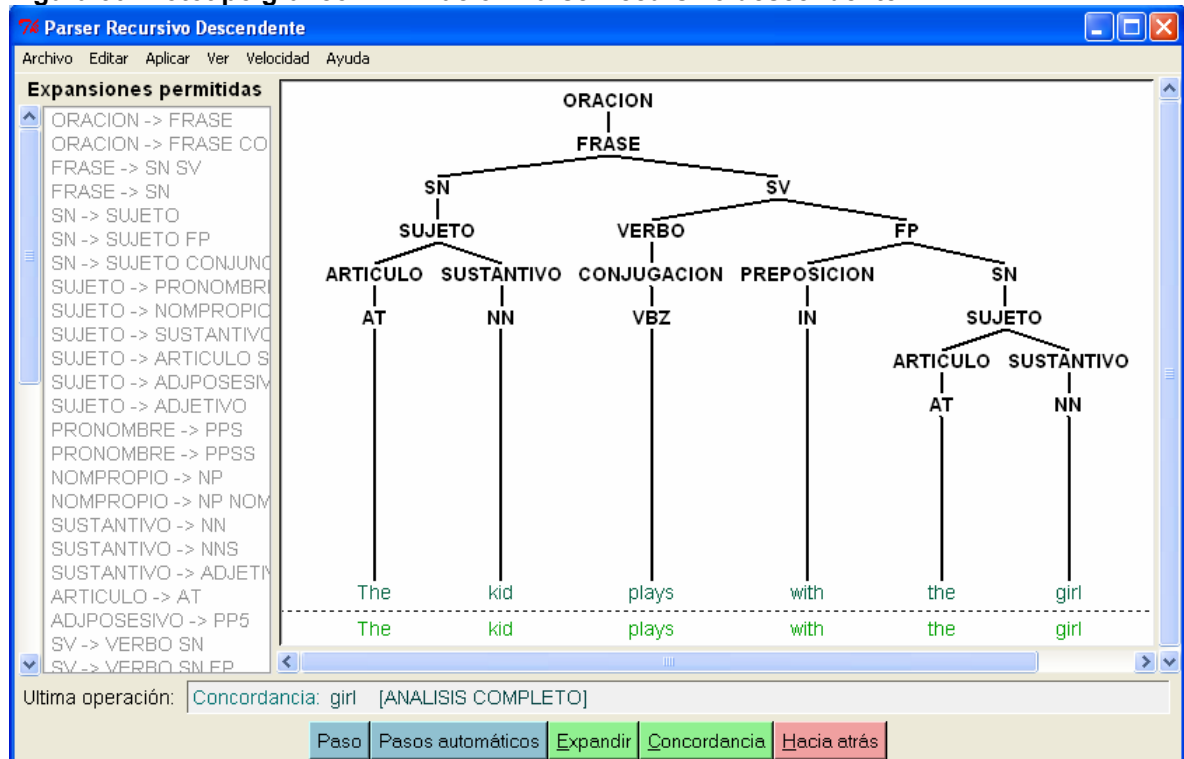
Nótese por ejemplo como para el token "*plays*" se tienen dos derivaciones morfológicas, al lado izquierdo de "==" se muestra la etiqueta que corresponde a la derivación y al lado derecho se muestra la información del morfema junto a su etiqueta que permite buscarlo en el diccionario. Siendo así, quiere decir que "*plays*" puede ser una conjugación en presente de tercera persona (*VBZ*) de la palabra *play*, la cual debe buscarse en el diccionario con la etiqueta *VB* (verbo) o que también puede ser un sustantivo plural derivado de "*play*" que se encuentra en el diccionario clasificado como *NN* (sustantivo singular).

Para el análisis sintáctico hay dos opciones, visualizar una animación que en base a la gramática muestra paso a paso como se realiza el análisis sintáctico o

visualizar el árbol que se genera después de realizar el análisis sintáctico, el cual asigna a cada palabra su correspondiente clasificación sintáctica.

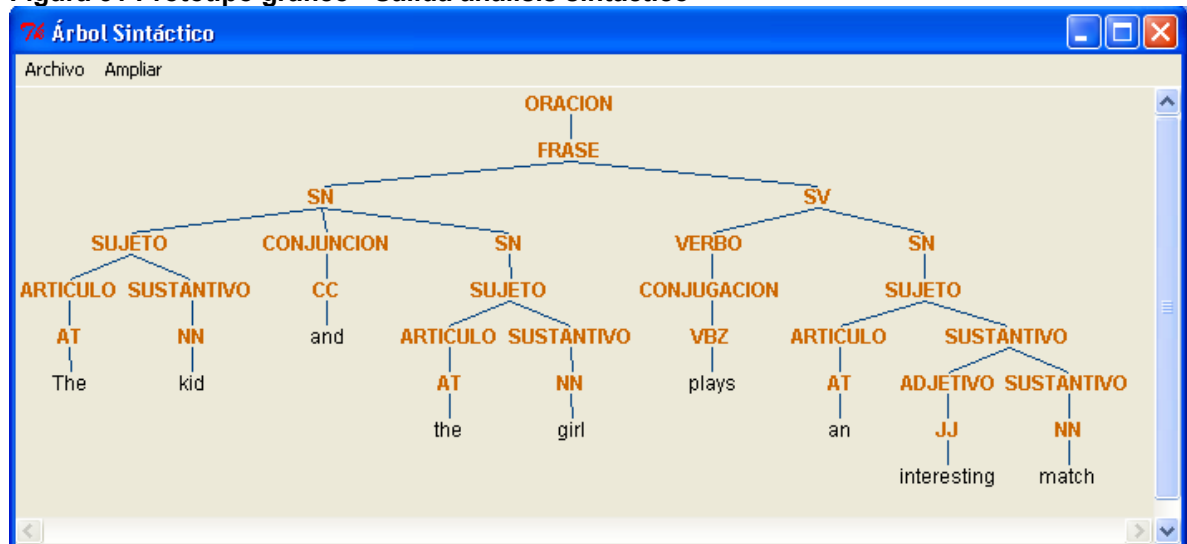
Si se oprime el botón de animación se desplegará una ventana con diferentes opciones para visualizar el análisis recursivo descendente paso a paso obteniendo al final un árbol sintáctico, esta ventana se observa en la Figura 30 y posee su propia ayuda:

Figura 30 Prototipo gráfico - Animación Parser recursivo descendente



Si se oprime el botón de árbol en el análisis sintáctico, se desplegará una ventana con el árbol sintáctico, es decir, a cada palabra se la clasifica sintácticamente de acuerdo a la gramática establecida para la formación de oraciones en presente simple, la ventana que se despliega se observa en la Figura 31:

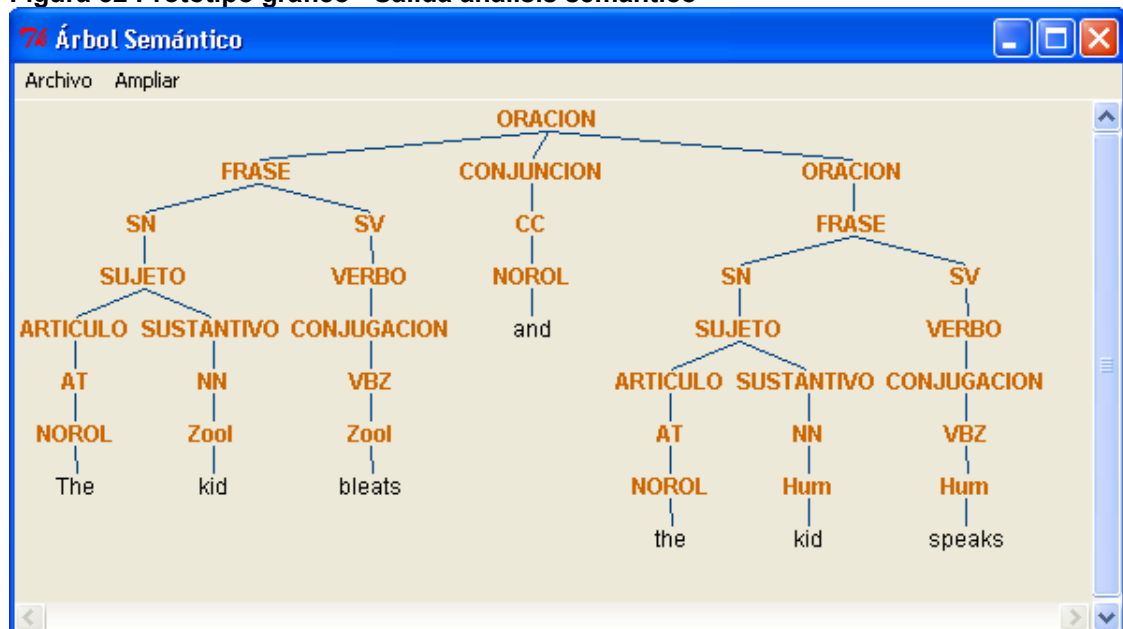
Figura 31 Prototipo gráfico - Salida análisis sintáctico



Si la oración de entrada no se encuentra bien formada de acuerdo a la gramática definida, saldrá una ventana de error indicándolo.

Al oprimir el botón de análisis semántico se desplegará una ventana con el resultado de dicho proceso como se observa en la Figura 32, el resultado es un árbol con la asignación de los roles semánticos para cada palabra y el acumulado de los procesos anteriores, así:

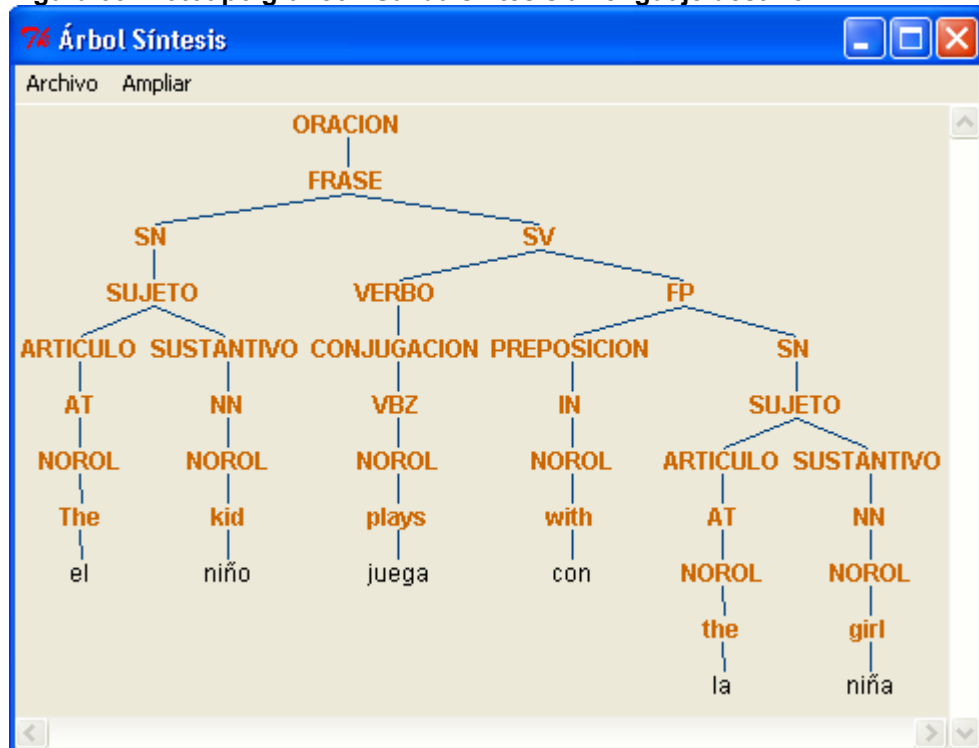
Figura 32 Prototipo gráfico - Salida análisis semántico



Igual que en el caso anterior, si la oración de entrada no se encuentra bien formada, saldrá una ventana de error indicándolo.

Si se oprime el botón de síntesis al lenguaje destino se desplegará una ventana con el resultado de dicho proceso como se observa en la Figura 33, el cual es un árbol con la traducción adecuada para la oración y el acumulado de los procesos anteriores, así:

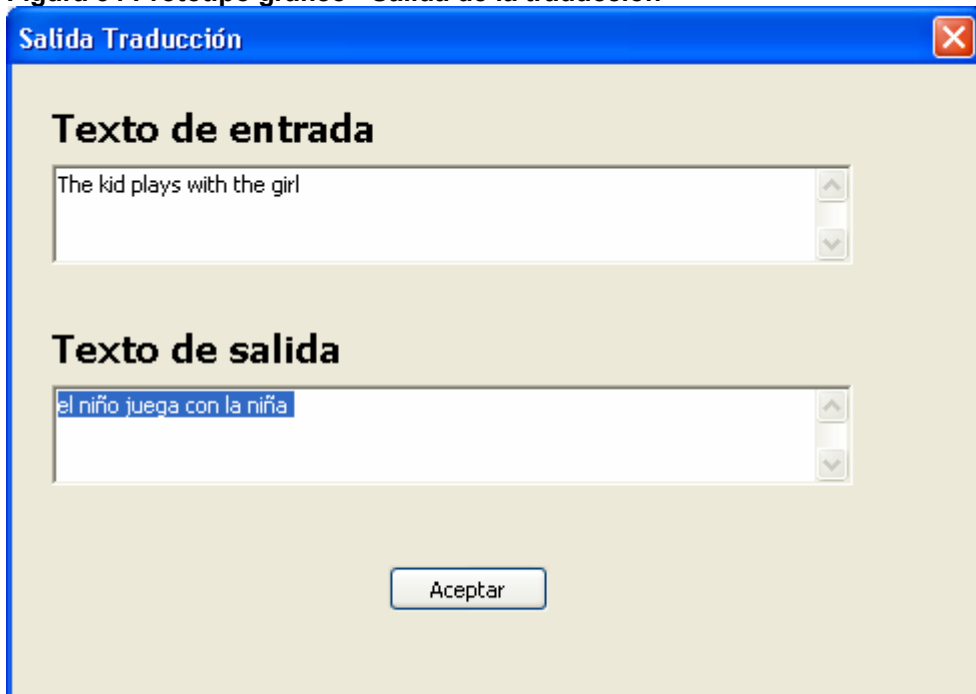
Figura 33 Prototipo gráfico - Salida síntesis al lenguaje destino



Si la oración de entrada no se encuentra bien formada, saldrá una ventana de error indicándolo.

Si se oprime el botón de salida se desplegará una ventana con el resultado de la traducción, como se observa en la Figura 34:

Figura 34 Prototipo gráfico - Salida de la traducción



Salida Traducción

Texto de entrada

The kid plays with the girl

Texto de salida

el niño juega con la niña

Aceptar

Si la oración de entrada no se encuentra bien formada según las reglas gramaticales definidas, entonces en lugar de la traducción de salida se mostrará el texto:

ERROR TRADUCTOR! No se ha podido generar una salida válida, posiblemente la oración no se encuentra bien formada.

11. APORTES

11.1. APORTES DEL DESARROLLO

- Se ha realizado una importante recopilación y documentación del estado del arte de la traducción automática. Dicho trabajo es amplio, conciso y está traducido al español. La historia recopilada permite conocer el trascender de la traducción por medio de la computadora, así como los intentos de transmitir ideas en diversos lenguajes. De igual forma brinda un panorama de las metodologías aplicadas actualmente para resolver el problema de la traducción automática, permitiendo de esta forma, ser un punto de partida para las investigaciones subsiguientes.
- El prototipo de tokenizador implementado y su diseño se realizaron con un enfoque que no se encuentra documentado en ninguno de los trabajos consultados. Acepta una gran cantidad de patrones, lo cual hace que el dominio de palabras y formatos de escritura sea bastante amplio. Aunque se hizo uso de la librería `tokenize` de *NLTK LITE*, en realidad son las expresiones regulares implementadas las que realizan la tarea primaria de tokenización.
- Se diseñó un modelo de diccionario que facilita la utilización de este en el proceso de traducción. Para esto se propone una estructura basada en tablas hash anidadas, logrando de esta manera un acceso rápido y sencillo a nivel de usuario, así como una estructura jerárquica para realizar la búsqueda. Como entrada, se tiene un archivo de texto, el cual tiene una estructura similar a los diccionarios de papel, permitiendo una fácil actualización. Adicional a esto, el diseño de diccionario puede manejar información de subcategorización y restricciones selectivas, información fundamental en los procesos de análisis sintáctico y semántico. Dicho diseño, además hace uso del *Brown Corpus Tag Set* permitiendo de esta manera contar con una estandarización en el etiquetamiento de las diferentes categorías de palabras.
- Se ha implementado un prototipo de motor de inferencia para el tratamiento de ambigüedades semánticas en la traducción automática, permitiendo mostrar la funcionalidad de este proceso específico en la generación de traducciones que tengan sentido para un ser humano, esto podría solventar el principal defecto que tienen los sistemas de traducción de mayor uso, que en la mayoría de los casos no tienen en cuenta el análisis semántico.

- Se realizó un conjugador de verbos que hace uso de etiquetamiento de verbos con conjugación similar, para conjugar correctamente en presente simple y en gerundio cerca de ocho mil verbos de la lengua española.
- Se propuso un algoritmo para realizar la conjunción de pronombres.
- Este trabajo ha aportado nuevos enfoques, que aunque a baja escala, permiten demostrar que es posible la generación de traducciones inteligibles por medio de una computadora.
- Los prototipos implementados son modulares y están ampliamente documentados. La documentación es tan exhaustiva que se ha documentado cada línea de código, logrando de esta manera tener un código ampliamente explicado, lo cual permite su fácil reutilización, aspecto muy importante en el desarrollo de software.
- En cada uno de los capítulos dedicados a las etapas del proceso de traducción se realizó una recopilación teórica y una implementación de un prototipo para mostrar la funcionalidad de dicha teoría. Enlazando la teoría y la práctica, se realizó para cada uno de dichos capítulos un modelamiento respectivo. Este modelamiento es sencillo y abarca el problema de manera general, permitiendo tener modelado el comportamiento de un sistema de traducción automática. de tal forma, se puede llegar a programar los prototipos en cualquier lenguaje de programación.
- Después de un año y de investigación en el desarrollo del presente trabajo no se encontró en Colombia una tesis dedicada al área de la traducción automática, aunque esto no necesariamente signifique que no hay aportes al respecto, este trabajo pretende colocar un grano de arena para que interesados en todo el país y en general los investigadores hispanoparlantes interesados en el área encuentren líneas de investigación interesantes y prometedoras relacionadas con el tema. Ya que si bien existe mucha documentación la mayor parte de ella está en inglés.
- Este trabajo es el primero que aborda el tema del procesamiento del lenguaje natural en la Universidad Tecnológica de Pereira. Seguramente será el punto de partida que impulse la creación de nuevos grupos de investigación interdisciplinarios en esta institución. Involucrando, dadas las características del tema abordado, a varios programas y facultades.

- El espectro de aporte del presente trabajo no abarca únicamente a los programas relacionados con la informática, sino también a aquellos dedicados al estudio del lenguaje.
- Para la gestión de las actividades, control de documentos e implementación de prototipos se hizo uso de herramientas de software libre. Para reivindicar los beneficios tomados de la comunidad de software libre, bajo esta misma modalidad se pondrán a disposición de todo el mundo los prototipos generados para que toda la comunidad interesada tenga una base de trabajo y el desarrollo escale a un sistema más robusto.
- En el desarrollo del trabajo se hizo un amplio estudio de documentos con mucha validez como tesis doctorales, papers, investigaciones, herramientas relacionadas con el proceso, etc. Documentación que ha sido organizada de acuerdo a cada tema e indexada para una búsqueda más eficiente. Esto permitirá que quienes estén interesados en continuar por esta línea de investigación tengan información confiable como base para su estudio.
- Para cada una de las etapas del proceso de traducción el lector podrá encontrar decenas de tesis doctorales abordando el tema, pero pocas de ellas divulgan una implementación de dicha teoría. Este trabajo permite conocer la implementación de prototipos a partir de una base teórica, dichos prototipos, así mismo están completamente documentados y libres.
- La documentación de *API's (Application Program Interface)* se generó en *HTML* haciendo uso de la herramienta *Epydoc*, que permite, a partir de la documentación contenida en el código fuente producir una serie de archivos *HTML*. Este aporte es, para los usuarios y continuadores del trabajo, fundamental, ya que conociendo las *API's* del presente desarrollo pueden usarlo y seguir desarrollando.
- Durante la realización de este proyecto se creó un grupo de investigación llamado *Grupo de Traducción Automática*, el cual se encuentra adscrito al semillero de investigación *PULPA* y tiene el correspondiente registro *GRUPLAC* de *COLCIENCIAS*. De igual forma los desarrolladores del proyecto están registrados en el *CVLAC*.
- Se establecieron contactos con investigadores importantes en el ámbito de la traducción automática a nivel mundial, como *John Hutchins*, presidente de la *EAMT* y autor del libro "*Machine Translation: Past, Present, Future*", guía indiscutible para cualquier trabajo de traducción automática, *Steven Bird* y *Edwar Loper* de la universidad de Pensilvania, desarrolladores de la herramienta *NLTK*,

Ismael Olea, representante de la comunidad de software libre en español, *Jesús Vilares* y *Jorge Graña*, estos últimos doctores en ciencias de la computación con tesis doctorales relacionadas con traducción automática.

11.2. APORTES EN CUANTO A LA METODOLOGÍA DE TRABAJO

- Para realizar el presente trabajo, se generó un documento de plan de negocios para la creación de un grupo de investigación de traducción automática. Dicho documento sirve de apoyo para grupos de investigación que deseen conformarse y evaluar la viabilidad de un proyecto y el presupuesto e infraestructura necesarios para hacerlo. Dicho documento se incluirá como anexo digital, esto debido a su volumen y que no se relaciona directamente con los objetivos del presente proyecto.
- Para la puesta en marcha y control adecuado del proyecto, se hizo uso de la herramienta dotProject. Esta es una herramienta de colaboración (groupware) que permite a los integrantes de un proyecto planificar sus tareas, hacer control de las mismas, interactuar con los demás participantes, manejar versiones de documentos, etc. Los estudiantes que hacen tesis de grado o interactúan en un proyecto escasamente hacen uso de este tipo de herramientas que permiten que un proyecto se lleve a cabo de manera ordenada y documentada. Este proyecto es pionero en el uso de dicha herramienta en la Universidad Tecnológica de Pereira y en el programa de Ingeniería de Sistemas y Computación, dando pie a que el grupo de investigación al cual se encuentra adscrito (Semillero de Investigación PULPA) lo haya adoptado como herramienta de trabajo para los proyectos que allí se desarrollen en el futuro.
- Una de las tareas mas tediosas al escribir trabajos de grado o de otro tipo es la de documentar de acuerdo a las normas técnicas colombianas, las referencias bibliográficas consultadas. Para este proyecto se hizo uso de la herramienta Wikindex, que permite administrar en línea (en el mismo momento en que se consultan) la información concerniente a dichas referencias. Gracias a esto, quienes realicen un trabajo escrito cuentan con un elemento que les permite generar automáticamente las referencias bibliográficas que cumplan con las normas y además, para quienes trabajen en esta línea de investigación en futuros proyectos se ha dejado un legado de referencia documental como base de estudio.
- Se elaboró una plantilla para el software MS-Word que contiene la base para generar documentos escritos aplicando las normas ICONTEC colombianas vigentes, la cual permite dar formato estándar a todo el escrito y permite a quienes generan la documentación de sus proyectos concentrarse más en el

fondo que en la forma de sus trabajos. Esta plantilla también se incluirá como anexo digital.

12. CONCLUSIONES

- Se logró hacer una recopilación amplia y concisa del estado del arte de la traducción automática. Se pudo conocer, a partir de dicha investigación, la historia de la traducción automática, así como los desarrollos que han existido en el área. Adicional a esto se pudieron identificar los avances actuales en el tema.
- Se pudo verificar que la traducción automática es un tema complejo y de difícil implementación. Esto permite, de igual manera, explicar por qué a pesar de ser uno de los temas de investigación más antiguos de la ciencia de la computación, los avances no tengan resultados satisfactorios desde el punto de vista de la obtención de traducciones automáticas comparables a las de los seres humanos.
- A pesar de identificar el alto grado de dificultad que tiene el proceso de traducción automática, se lograron superar los objetivos, pudiendo enlazar todos los prototipos, para así obtener traducciones inteligibles.
- Se identificaron claramente cada una de las etapas asociadas al proceso de traducción automática, sus principales características y reglas asociadas a cada una de las mismas. De igual forma, los factores que deben ser considerados para lograr obtener un resultado satisfactorio.
- Se realizaron prototipos para cada una de las etapas del proceso de traducción y se mostró su funcionalidad con ejemplos prácticos.
- En la etapa de preprocesamiento, se solucionaron los problemas relacionados con el filtrado del texto de entrada y la separación de contracciones, tal que no contenga espacios en blanco innecesarios, como tabulaciones y espacios redundantes e identificación de las palabras que han sido separadas por un guión y un salto de línea, además de eliminar las contracciones que no generan ambigüedades.
- El presente trabajo resuelve la mayoría de los problemas que se pueden presentar en el proceso de tokenización para el idioma inglés. El tokenizador contempla palabras, palabras con números, números, palabras separadas por guiones, palabras con contracciones, valores monetarios, porcentajes, fechas, horas, abreviaciones, nombres propios, signos de puntuación y palabras compuestas. Adicionalmente el manejo de las abreviaciones y las palabras

especiales en el archivo de excepciones permite tokenizar cualquier tipo de palabra que las expresiones regulares no puedan contemplar.

- Se diseñó un modelo de diccionario que facilita la utilización de éste en el proceso de traducción. Se propuso para su implementación, una estructura de tablas Hash anidadas, con lo cual se logran múltiples ventajas en comparación con otras soluciones.
- El analizador morfológico implementado etiqueta los tokens con el estándar de las etiquetas *Brown*, lo cual hace que su salida sea de igual forma estándar. Al igual que el tokenizador el diseño de este prototipo es utiliza un enfoque que no existe documentado por ningún autor. En este paso, se etiquetan gramaticalmente las palabras haciendo uso del diccionario.
- Con respecto al analizador sintáctico y el enfoque de estructura gramatical a usar, se optó por el enfoque de constituyentes como esquema, debido a la alta complejidad y el alto nivel de conocimiento lingüístico que requiere la construcción de un lexicón que involucre relaciones de dependencias en caso de adoptarse el enfoque de dependencias.
- Se implementó un prototipo de motor de inferencia para el tratamiento de ambigüedades semánticas en la traducción automática, específicamente atendiendo problemas relacionados con el conocimiento de tipo semántico, permitiendo mostrar la funcionalidad de este proceso específico en la generación de traducciones que tengan sentido para un ser humano.
- En el proceso de síntesis al lenguaje destino, se realizó un conjugador de verbos que hace uso de etiquetamiento de verbos con conjugación similar, para conjugar correctamente en presente simple y en gerundio cerca de ocho mil verbos de la lengua española. Se realizaron algoritmos para formación de palabras en plural y cambio de género de un adjetivo. De igual forma, se propuso un algoritmo para realizar la conjunción de pronombres.
- Se construyó una interfaz gráfica, que permite hacer uso de una manera mucho más sencilla del prototipo de traductor.
- Enlazando la teoría y la práctica, se realizó para cada una de las etapas del proceso de traducción un modelo. Este modelo es sencillo y abarca el problema de manera general, permitiendo programar los prototipos en cualquier lenguaje de programación.

- Mediante los ejemplos con que se probaron los prototipos, se pudo mostrar la viabilidad del desarrollo de programas que permitan realizar las etapas del proceso de traducción automática.
- Se lograron resolver problemas de orden semántico independiente del contexto, los cuales, en la mayoría de los casos, no son resueltos por los traductores comerciales, tampoco, este tipo de análisis es efectuado por los traductores disponibles gratuitamente en Internet, es decir, la traducción obtenida supera semánticamente la que se obtiene con estos últimos traductores.
- El desarrollo de los prototipos fue realizado en el lenguaje de programación Python, el cual es un lenguaje de scripts de fácil entendimiento, con un excelente manejo de cadenas de caracteres (fundamental para desarrollos relacionados con el tema). La sencillez del lenguaje permite que los prototipos desarrollados sean a la vez sencillos y fáciles de entender, sumado a esto las pocas líneas de código en que se desarrollaron.

13. RECOMENDACIONES

- Los prototipos desarrollados no hacen uso de excepciones, así que cualquier error producido por inconsistencias en uno de los archivos de trabajo (diccionario, excepciones, contracciones, etc.) puede dar pie a que los prototipos fallen. Es necesario que se involucren a los prototipos el adecuado manejo de excepciones dando una base más robusta para la construcción de un prototipo a mayor escala.
- Es necesario fomentar en el programa de Ingeniería de Sistemas y Computación de la Universidad Tecnológica de Pereira el uso de herramientas que permitan a los integrantes de un proyecto concentrarse más en la planificación, desarrollo y control de las actividades y no en los aspectos de forma que pueden llegar a consumir mucho tiempo si no se cuenta con una herramienta que permita automatizar estas actividades.
- Para la continuidad del proyecto debería crearse un grupo interdisciplinario de traducción automática, tal que el desarrollo no sea exclusivo de Ingeniería de Sistemas, ya que un tema tan complejo como la traducción automática exige la atención de personas con conocimiento del lenguaje.

14. TRABAJOS FUTUROS

- El módulo de análisis morfológico puede ser muy complejo. Si bien en el presente trabajo se mostró la funcionalidad de éste, se hace necesario ampliar el espectro de morfemas a ser tenidos en cuenta.
- El análisis sintáctico de este desarrollo sólo contempla un rango reducido de frases que pueden conformarse para el idioma inglés. Se hace necesario ampliar dicho rango para una profundización en la solución del problema.
- El motor de inferencia usado en el análisis semántico muestra la manera de eliminar ambigüedades de orden semánticas. Dicho motor de inferencia puede ser mejorado para reducir de una manera más profunda los roles ambiguos que pueden presentarse en una palabra, teniendo en cuenta la relación entre los roles de las palabras que constituyen una frase.
- El conjugador de verbos puede ser expandido de manera que pueda conjugar en los tiempos que no fueron tenidos en cuenta en el presente trabajo.
- Los modelos elaborados para cada prototipo, permiten que los mismos sean programados en cualquier lenguaje de programación, así se podrían optimizar algunas de las etapas programándolas en un lenguaje que cumpla las características que dicha etapa exija.
- El diccionario con que cuenta el prototipo es limitado, un proyecto podría tener como objetivo principal ampliar el diccionario de tal forma que las traducciones no se vean tan limitadas. Una propuesta interesante podría ser que la fuente del diccionario fuera uno de los servicios de diccionarios a través de Internet, de tal forma que la actualización del diccionario se más eficiente.
- El prototipo funciona como aplicación de escritorio que necesita ser instalada, sería de gran utilidad que pudiera desarrollarse una aplicación Web a partir de esta y ofrecerlo como un servicio en Internet.
- Este trabajo puede ser tomado como base para la elaboración de un libro acerca de la traducción automática y su implementación, pues enlaza la teoría y la práctica de una manera clara y sencilla, esto, al estilo del autor *Andrew S. Tanenbaum* en su libro *Sistemas operativos: diseño e implementación*, cuya implementación básica de un sistema operativo (*Minix*) dio pie al desarrollo de uno de los sistemas operativos más usados y más robustos en la actualidad (*Linux*).

- El espectro de aporte del presente trabajo no abarca únicamente a los programas relacionados con la informática, sino también a aquellos dedicados al estudio del lenguaje. De tal forma, este desarrollo puede servir de base para la creación de un grupo interdisciplinario de la traducción automática.

BIBLIOGRAFÍA

A Digital Archive of Research Papers in Computational Linguistics. 22 Jan 2006.
<<http://acl.ldc.upenn.edu/>>

ABAITUA, Joseba J. A. "Traducción automática: Presente y Futuro: Cuello de botella de la sociedad de la información". 1 Jun 2005.
<http://www.foreignword.com/es/Technology/art/Abaitua/Abaitua_1.htm>

American Demographics. 1 Jan 2005.
<<http://www.demographics.com/publications/fc/97%5Ffc/9712%5Ffc/f971201.htm>>

ARNOLD, Doug D. A. "Machine Translation: an Introductory Guide." NCC BlackWell (1994).

Automated Real-Time Translation. 2001. 1 Jun 2005.
<<http://www.sdl.com/es/localization-information/white-papers-articles/white-papers-list/white-papers-automated-real-time-translation.htm>>

Bases de datos y bases de conocimiento. 1 Jun 2005.
<<http://elies.rediris.es/elies18/522.html>>

BEARE, Kenneth. "How many people learn English globally?". 15 Mar 2005.
<<http://gda.utp.edu.co/traductor/Documentacion/EstadoArte/Articulos/articulo9.htm>>

CIA World Fact Yearbook. 1 Jun 2005.
<<http://www.odci.gov/cia/publications/factbook>>

Codes for the Representation of Names of Languages. 27 Jan 2006.
<<http://www.loc.gov/standards/iso639-2/engl angn.html>>

Documentación de Python: Tutorial de Python en español. 7 Feb 2006.
<<http://pyspanishdoc.sourceforge.net/>>

ERICHSEN, Gerald. "Spanish Facts and Stats". 18 Mar 2005.
<<http://gda.utp.edu.co/traductor/Documentacion/EstadoArte/Articulos/articulo7.htm>>

Elementary Language Processing: Tokenizing Text and Classifying Words. 2005. 1 Dec 2005.

[<http://nl tk.sourceforge.net/tutorial/tokenization/nochunks.html>](http://nl tk.sourceforge.net/tutorial/tokenization/nochunks.html)

Enterprise-Wide Cross-language Communication. 17 Apr 2005.

[<http://gda.utp.edu.co/traductor/Documentacion/Articulos/Otros/articulo3.jsp.htm>](http://gda.utp.edu.co/traductor/Documentacion/Articulos/Otros/articulo3.jsp.htm)

Esperanto: la lengua planetaria. Rumbo a la sociedad-mundo. 1 Jun 2005.

[<http://www.monografias.com/trabajos18/esperanto/esperanto.shtml>](http://www.monografias.com/trabajos18/esperanto/esperanto.shtml)

Esquemas Lógicos. 1 Jun 2005.

[<http://elies.rediris.es/elies9/4-3-1.htm>](http://elies.rediris.es/elies9/4-3-1.htm)

FBI no está traduciendo todo el material terrorista. Terra 2004. 19 Mar 2005.

[<http://terra.com.pr/actualidad/articulo/html/act184082.htm>](http://terra.com.pr/actualidad/articulo/html/act184082.htm)

GALICIA HARO, Sofia Natalia. "Análisis sintáctico conducido por un diccionario de patrones de manejo sintáctico para lenguaje español." Doctoral Thesis.

INSTITUTO POLITÉCNICO NACIONAL, 2000.

Global Internet Statics. 7 Feb 2006.

[<http://www.glreach.com/globstats/index.php3>](http://www.glreach.com/globstats/index.php3)

GREFENSTETTE, Gregory G. G. y TAPANAINEN, Pasi P. T. 1994. What is a Word, What is a sentence? Problems of Tokenization.

HANCOX, Peter P. H. "MORPHOLOGICAL ANALYSIS: Morphological processes". 1 Jun 2005.

[<http://www.cs.bham.ac.uk/~pjh/semla5/pt2/pt2_intro_morphology.html>](http://www.cs.bham.ac.uk/~pjh/semla5/pt2/pt2_intro_morphology.html)

HANCOX, Peter P. H. "MORPHOLOGICAL ANALYSIS: THE CLASSIFICATION OF MORPHOLOGICAL STRUCTURAL TYPES". 1 Jun 2005.

[<http://www.cs.bham.ac.uk/~pjh/semla5/pt2/pt2_intro_morphology.html>](http://www.cs.bham.ac.uk/~pjh/semla5/pt2/pt2_intro_morphology.html)

HERNANDEZ, Pilar P. H. "En torno a la traducción automática." El concepto de la traducción automática (2002).

HUTCHINS, John J. H. et al. EUROPEAN ASSOCIATION FOR MACHINE TRANSLATION, Compendium of Translation Software. 2005.

HUTCHINS, John J. H. Machine translation: Computer-based translation.

HUTCHINS, John. "The State of Machine Translation in Europe and Future Prospects". 16 Mar 2005.

<http://gda.utp.edu.co/traductor/Documentacion/EstadoArte/Articulos/articulo5.htm>

La traducción automática. 1 Jun 2005.

<http://www.dgbiblio.unam.mx/servicios/dgb/publicdgb/bole/fulltext/volIV3/traduccion.htm>

LONGLEY, Robert. "Nearly 1-in-5 Americans Speak Foreign Language at Home". 17 Mar 2005.

<http://gda.utp.edu.co/traductor/Documentacion/EstadoArte/Articulos/articulo8.htm>

MERRIMACK, N. H. M. "Enterprise Translation Server." Overview: How real-time translation works.

MORENO ORTÍZ, Antonio. Diseño e Implementación de un lexicón computacional para lexicografía y traducción automática. 9th ed. España: Estudios de Lingüística del Español, 2000.

MURRAY, Hubert Jr H. M. Methos for Satisfying the Needs of the Scientist and the Engineer for Scientific and Technical Communication.

PAPINENI, Kishore K. P. et al. 2001. Bleu: a Method for Automatic Evaluation of Machine.

PATTERSON, Wright W. P. Civil Engineering Center, Estimaciones de costos de traducciones. 1992.

PAVEL, Silvia S. P. y NOLET, Diane D. N. 2002. MANUAL DE TERMINOLOGÍA. Ministerio de Obras Públicas y Servicios Gubernamentales de Canadá.

RICHARDS, John. "The rules concerning the use of Apostrophes in written English". 28 Jan 2006. <http://www.apostrophe.fsnet.co.uk/>

RUBIN, G. M. and GREENE, B. B. 1981. Brown Corpus Tag Set. <http://www.comp.leeds.ac.uk/amalgam/tagsets/brown.html>: Automatic grammatical tagging of English. [Providence, R.I.: Department of Linguistics, Brown University].

SAG, IVAN A. and WASOW, TOMAS. Syntactic Theory: A Formal Introduction.

The Apostrophe. 28 Jan 2006.

[<http://owl.english.purdue.edu/handouts/grammar/g_apost.html>](http://owl.english.purdue.edu/handouts/grammar/g_apost.html)

UNESCO, Proyecto de recomendación sobre la promoción y el uso del plurilingüismo y el acceso universal al ciberespacio e informe del director general. 2001.

Universal Networking Language. 2005. 1 Jun 2005.

[<http://www.undl.org/unl sys/unl/unl2005/main.htm>](http://www.undl.org/unl sys/unl/unl2005/main.htm)

VILARES FERRO, Jesús. "Aplicaciones del procesamiento del lenguaje natural en la recuperación de información en español." Doctoral Thesis. UNIVERSIDAD DE LA CORUÑA, 2005.

Warren Weaver Memorandum. MT News International Jul 1999.

Web Site Translation Is Fastest Growing Segment Of Worldwide LANguage Translation Industry. 2002. 21 May 2005.

[<http://gda.utp.edu.co/traductor/Documentacion/General/Articulos/articulo1.jsp.htm>](http://gda.utp.edu.co/traductor/Documentacion/General/Articulos/articulo1.jsp.htm)

WILKINS, John J. W. "An Essay Towards a Real Character and A Philosophical Language". 1 Jun 2005. [<http://www.thoennes.com/language/wilkins.htm>](http://www.thoennes.com/language/wilkins.htm)

ÍNDICE

- ALGOL, 42
- ALPAC (Automatic Language Processing Advisory Committee), 42, 43
- ALPS, 44
- AltaVista, 46
- análisis morfológico, 21, 103, 111
- análisis semántico, 21, 141
- análisis sintáctico, 21, 125
 - ascendente, 130
 - descendente, 129
- AppTek, 47
- asimilación, 32
- átono, 104
- ATRIL, 45
- BABEL FISH, 46
- BLEU, 45
- caso, 107
- categoría, 92
- CE (Comunidad Europea), 19, 42, 43, 44, 46, 53
- clasificación semántica, 92
- clasificación sintáctica, 92
- contexto, 143
- contracciones, 20
- Danzin - informe, 44
- DÉJÀ-VU, 45
- derivación, 108
- diccionario, 21, 84
 - de papel, 85
 - importancia en la TA, 84
- diccionario electrónico, 49
- dirección de la traducción
 - bidireccional, 31
 - unidireccional, 31
- diseminación, 32
- EAMT, 47
 - Compendio, 47
- edición, 28
- estaciones de traducción, 52
- estructura gramatical
 - enfoque de constituyentes, 127
 - enfoque de dependencias, 126
- EUROTRA, 43, 44
- EWTranslate, 47
- filtro, 20

- formateo, 56
- frase, 19
- Fujitsu ATLAS, 47
- Fully Automatic High Quality Translation (FAHQT), 41, 42
- GACHOT, 46
- genero, 92
- género, 107
- GLOBALINK, 44
- grafos conceptuales, 36
- gramática
 - descriptiva, 125
 - prescriptiva, 125
- herramientas de alineamiento, 51
- herramientas de preedición, 52
- herramientas para el soporte de traducción, 49
- herramientas para soporte de localización, 50
- IBM, 44, 45
- IBM WebSphere, 47
- inflexión, 107
- inflexional, 112
- información de subcategorización, 86
- inteligencia artificial, 144
- intercambio, 32
- interfijo, 104
- interlingua, 33, 34
- interpretación, 25, 26
- ISO 639, 66
- KBMT, 37
- Kielikone, 47
- lengua franca, 18
- lenguaje aislado, 106
- lenguaje artificial, 69
- lenguaje destino, 24
- lenguaje fuente, 24
- lenguaje natural, 17, 36
- lenguajes aglutinantes, 106
- lexema, 103
- lexicón, 128
- LISP, 42
- localización, 27, 50
- LogoMedia Enterprise, 47
- Meaning-Text Theory, 126
- memoria de traducción, 44, 50
- METAL, 41, 43
- modo, 107
- monema, 103

- morfema, 103
 - cero, 105
 - libre, 104
 - trabado, 104
- morfología, 103
- MT Machine Translation), 17
- National Research Council, 42
- NEC, 44
- neologismo, 109
- NILS, 46
- Ntran, 44
- número, 107
- oración, 19
- palabra, 19, 70
- parser, 128, 130, 131, 132
- parsing, 128
- persona, 107
- Philips, 44
- PLN, 36
- PLN (Procesamiento de Lenguaje Natural), 107, 108, 110, 111, 128
- post-edición, 40
- pre-edición, 40
- prefijo, 104
- preprocesador
 - arquitectura, 56
 - interfaz, 67
 - uso del, 67
- preprocesamiento, 20, 56
 - contracciones, 59
 - específico, 64
 - filtro, 56
- procesamiento del lenguaje natural **PLN**, 36
- PROLOG, 44
- pronombres anafóricos, 143
- redes de marcos, 36
- redes IS-A, 36
- representación del conocimiento, 35, 141
 - del mundo real, 142, 143
 - esquemas de redes semánticas, 35
 - esquemas lógicos, 35, 36
 - pragmático, 142, 143
 - semántico, 142
- restricciones selectivas, 86
- revisión, 26
- rol, 92
- Rosetta, 44

- Sanyo, 44
- SDL Enterprise Translator, 47
- segmento fonémico, 105
- semi-afijo, 109
- servicio de traducción, 52
- significado, 103
- significante, 103
- sintagma nominal, 127
- sintagma verbal, 132
- sintaxis, 103
- síntesis al lenguaje destino, 22
- sistema de KBMT, 37
- sistema de manejo de terminología, 51
- sistemas bilingües, 31
- sistemas de memorias de traducción, 50
- sistemas de primera generación, 41
- sistemas de traducción directa, 33
- sistemas de transferencia, 33
- sistemas interlingua, 33
- sistemas multilingües, 31
- software de TA, 48
 - sistemas TA (Cliente/Servidor), 49
 - sistemas TA (Generales), 48
 - sistemas TA (Internet y Web), 49
 - sistemas TA (Uso doméstico), 49
 - sistemas TA (Uso profesional), 49
- STAR, 44
- sufijo, 104
- SYSTRAN, 41, 43, 46, 47, 53
- TA
 - basada en el conocimiento, 34
 - empírico, 34
 - simbólicos, 34
 - división de los sistemas de TA, 31
 - enfoques de la TA, 33
 - historia de la TA, 37
 - mercado de la TA, 53
- TA (Traducción Automática), 17, 109
- TAUM (Traduction Automatique de l'Université de Montréal), 43
- terminología, 27
- texto destino, 17
- texto origen, 17
- tiempo, 107
- tipo, 71, 117
- tipología, 18
- token, 69, 71

tokenización, 20, 69, 70, 71, 75
TRADOS, 44
traducción, 17, 24, 26
traducción asistida, 31
traducción audiovisual, 29
traducción humana asistida por computador, 31
traducción totalmente automática, 31
TRANSIT, 45
TranslationManager, 44
Translator's Workbench, 44
TranSmart, 47
TranSphere, 47
UMIST, 44
UNESCO, 221
unidad cognitiva, 19
unidades de traducción, 19
unidades significativas, 103
Web, 28, 32, 50, 52, 53

ANEXOS

ANEXO A TENDENCIAS DE IDIOMAS Y CULTURAS

HECHOS Y ESTADÍSTICAS ACERCA DEL IDIOMA ESPAÑOL

Para los estadounidenses el aprender español se ha venido convirtiendo en una necesidad ya que existen múltiples factores que han ido sumando a la importancia de aprender esta lengua, entre ellos se destacan que el español es la primera lengua más común de América, además de los hispano parlantes en España, en Estados Unidos, las Filipinas, y África pueden también encontrarse muchas personas que manejan éste idioma.

Hay además otros hechos importantes sobre español que pueden ser interesantes:

El español, junto con francés, es la lengua oficial de Guinea ecuatorial, haciéndole el único país en África con una presencia española oficial, aunque el español también es hablada por algunos en Marruecos. El nombre oficial del país es República de Guinea Ecuatorial¹⁴⁰.

Otros países o áreas semi-autónomas con las poblaciones de habla hispana significativas incluyen Andorra, la Argentina, Belice, Bolivia, Chile, Colombia, Costa Rica, Cuba, la República Dominicana, Ecuador, El Salvador, Francia, Gibraltar, Guatemala, Honduras, México, Nicaragua, Panamá, Paraguay, Perú, las Filipinas, Puerto Rico, Uruguay, los Estados Unidos y Venezuela¹⁴¹.

Casi el 30 por ciento de los residentes de España tienen como lengua materna otra diferente al español, aunque la mayoría usan el español como segunda lengua. Los idiomas de España incluyen al catalán (Cerca de un 12 por ciento de la población la hablan como primera lengua, y la hablan aún más como segunda lengua), el gallego (8 por ciento de la población) y el vasco (un poco más del 1 por ciento)¹⁴².

¹⁴⁰ "CIA World Fact Yearbook". 1 Jun 2005. <<http://www.odci.gov/cia/publications/factbook>>.

¹⁴¹ "Ethnologue". 1 Jun 2005. <<http://www.ethnologue.com>>. Op. Cit.

¹⁴² Ibid.

En 1998, Estados Unidos tenía la quinta población hispánica más grande, cerca de 30 millones de personas. De ellos, dos tercios remonta sus raíces a México y 86 por ciento dicen que el español es su primera lengua¹⁴³.

Solamente California tiene 5.5 millones de personas que hablan español en el país. Otros estados con altas poblaciones de habla hispana incluyen Tejas (3.4 millones), Nueva York (1.8 millones) y Florida (1.5 millones)¹⁴⁴.

Cerca de 9 por ciento de la gente que utiliza el Internet habla español, haciéndola la tercera lengua en la comunidad del Internet, después del inglés con 35.2 por ciento y china con 13.7 por ciento. Muy cerca está el japonés con 8.4 por ciento, seguido por el alemán con 6.9 por ciento¹⁴⁵.

El índice de natalidad en los hablantes de la lengua hispana es más alto que el de las personas que tienen el inglés como primer lengua, el español puede fácilmente permanecer firmemente en el segundo puesto en los años por venir. Además, si las economías de los países latinoamericanos mejoran, el español podría ganar importancia en el comercio y en las comunicaciones a nivel mundial.

INFORME DEL CENSO «USE AND ENGLISH-SPEAKING ABILITY»¹⁴⁶

Casi 47 millones de personas, la quinta parte de los residentes de los Estados Unidos, mayores a 5 años, reportaron hablar regularmente un idioma extranjero en el país en el año 2000, según la oficina de censo. Esto representa un aumento de 15 millones de personas desde el censo de 1990.

El nuevo informe del censo, *Use and English-Speaking Ability 2000*, dijo que el 55 por ciento de la gente que habla un idioma diferente al inglés, también dijeron hablar el inglés muy bien, de quienes solamente hablan inglés, el 92 por ciento decían no tener ninguna dificultad en hablar el inglés.

El español sigue siendo la lengua más comúnmente hablada a excepción del inglés, los hispano parlantes aumentaron de 17.3 millones en 1990 a 28.1 millones en el 2000, eso un incremento del 62 por ciento. Más de la mitad de los hispanohablantes manifestaron manejar el inglés muy bien.

¹⁴³ "American Demographics". 1 Jan 2005. <<http://www.demographics.com/publications/fc/97%5Ffc/9712%5Ffc/f971201.htm>>.

¹⁴⁴ Ibid.

¹⁴⁵ 2006-02-07 16:22:34 "Global Internet Statics"

¹⁴⁶ En español: Uso y capacidad del habla inglesa

TA EN LA UNIÓN EUROPEA

La Unión Europea es uno de los más viejos usuarios de la TA (además de la Fuerza Aérea estadounidense), y es probablemente el más grande usuario de esta tecnología.

Allí, los sistemas de TA son usados para producir traducciones que son editadas por traductores con el ánimo de obtener textos de buena calidad (publicable). En muchos casos, traductores de la CE usan las salidas de los TA como borradores de una traducción.

En la mayoría de los casos, la TA es usada para la documentación interna. En pocas ocasiones es usada por traductores cuando se pretende publicar los documentos externamente. Sin embargo, los traductores no son los principales usuarios del *Systran* (Senez¹⁴⁷, D. (1995): *Developments in Systran, Aslib Proceedings*). Los principales usuarios son los administradores de la comisión, quienes reciben cientos de documentos de los cuales necesitan saber de qué tratan y así determinar si es necesario solicitar la traducción de un traductor humano. Otro uso que le dan estos administradores es para escribir borradores en un idioma no nativo en el cual no son muy fluidos.

Un uso más reciente de los sistemas de TA es el que le dan los intérpretes de la comisión, para traducir textos que ellos saben van a ser reportados en las sesiones del parlamento Europeo u otra reunión, entonces se basan en estos documentos para su interpretación.

La comisión cuenta ahora con versiones de *Systran* para muchos pares de idiomas: inglés-francés fue el primero, luego francés-inglés, inglés-italiano, seguidos por otros pares de que involucran al alemán, holandés, español, griego y portugués. Por obvias razones se involucraron primero los idiomas más comunes. En años venideros los idiomas del oriente europeo serán añadidos, comenzando desde ya el trabajo para el checo y el polaco.

La demanda de TA dentro de la Comisión ha crecido en un 500% en menos de 10 años (desde mediados de los 90's) y más de un 20% anualmente. Se espera que esta tasa siga creciendo en los próximos años.

Por otro lado, la calidad no ha respondido tan rápido, pero se ha detectado dentro de la Comisión que entre más se restrinja el dominio en cuanto al tema o tipo de documento, se pueden lograr sustanciales mejoras en cuanto a la calidad.

¹⁴⁷ SENEZ, Dorothy. Commission of European Communities Developments in SYSTRAN.

ANEXO B PROBLEMAS DE COMUNICACIÓN EN GRANDES EMPRESAS

La confiabilidad de la comunicación de una empresa y la colaboración está cobrando alta importancia debido a los cambios de las políticas extranjeras, el crecimiento del Internet y el desarrollo de las nuevas tecnologías que están rompiendo rápidamente barreras a la extensión del mercado internacional. Los negocios también han experimentado un aumento rápido en la velocidad de la comunicación y del intercambio de la información global. Un negocio acertado entiende que cualquier éxito de mercado extranjero requiere más que la traducción simple de la literatura de comercialización. Hoy, el éxito en una escala global requiere una comprensión de las diversas culturas y de las necesidades y requisitos de sus mercados.

Para cualquier organización entender y responder a los cambios de necesidades de los mercados extranjeros es altamente importante, para esto, deben tener las herramientas para comunicarse en una escala global. Las organizaciones que pueden efectuar una comunicación en tiempo real no importando el tipo de idiomas tienen una ventaja competitiva.

Hasta este momento, la comunicación, la colaboración y el comercio electrónico en tiempo real a través del lenguaje no eran posibles, pero con la introducción del software de TA, cualquier organización global puede abrir fácilmente sus canales de comunicaciones para permitir que sus empleados se comuniquen y compartan ideas en varios idiomas.

El software de traducción sistematizada entrega borradores de alta calidad de las traducciones en tiempo real y automáticamente, que proveen a los individuos y profesionales el significado del documento original no importando la lengua en que se encontraba de modo que puedan determinar la importancia de los mismos o de su negocio.

Los siguientes métodos estándares de comunicación corporativa se pueden abrir fácilmente con el software de TA.

COMUNICACIÓN MULTILINGÜE VÍA INTRANET

La rápida aparición y la adopción del Intranet por las empresas se han incrementado altamente los últimos dos años. La mayoría de las organizaciones medianas y grandes despliegan niveles de servicio de Intranet a sus empleados. El propósito primario de un Intranet es permitir a cada oficina dentro de una organización global comunicar simultáneamente noticias y la información a cada empleado. La creación de noticias e información de alto valor para los empleados

en la Intranet requiere que cada empleado pueda leer y responder en una lengua que puedan entender fácilmente.

Con el software de TA, las organizaciones pueden fácil y rentablemente ampliar su Intranet para permitir el acceso multilingüe e instantáneo al contenido necesario. Desde su Intranet, los empleados podrían solicitar la traducción de cualquier página en la lengua de su opción. En cuestión de segundos, el empleado recibirá un borrador de alta calidad de la traducción del documento original el cual estaba en una lengua extranjera y podrá determinar su importancia para él y su departamento. La lengua no sería entonces una barrera para recibir noticias e información.

COMUNICACIÓN MULTILINGÜE VÍA E-MAIL

Hoy, el *e-mail* es el método preferido de comunicación para la mayoría de las personas de negocios. El *e-mail* es una herramienta rápida, rentable de comunicación el cual está disponible las 24 horas del día en casi cada país del mundo. La única barrera global para comunicarse vía *e-mail* es la lengua. Es posible emplear un traductor humano para traducir los *e-mail* de la lengua extranjera, pero para el 99 por ciento de negocios el costo y el tiempo de traducción son inaccesibles.

Integrando un software de TA en un sistema de *e-mail*, cada empleado puede traducir inmediatamente mensajes de una lengua extranjera a una lengua de su opción. Además, los empleados también tienen la opción de escribir un *e-mail* en su lengua materna y automáticamente traducirlo en la lengua del receptor y después enviarlo. Para muchos empleados, esto les permite comunicarse directamente con los colegas extranjeros rápidamente, simplificando el procedimiento de compartir información y la toma de decisiones.

COLABORACIÓN MULTILINGÜE VÍA GROUPWARE¹⁴⁸

El software *Groupware*, que incorpora *e-mail* y charla en línea, se está convirtiendo rápidamente en el método preferido para que las corporaciones permitan a sus empleados internacionales colaborar en proyectos de oficinas alrededor del mundo. Sin embargo, muchas organizaciones no pueden tomar ventaja completa de la colaboración internacional usando el *groupware* debido a las barreras lingüísticas.

¹⁴⁸ En español: Grupo de trabajo

Para muchas organizaciones, solamente empleados internacionales que manejen el idioma inglés pueden unirse a un equipo global. Los empleados que no pueden comunicarse en inglés son incapaces o renuentes a compartir sus ideas o conocimiento con sus compañeros de trabajo.

Integrando traducciones en tiempo real en un uso del *groupware*, los empleados internacionales pueden poner sus ideas y conocimiento a disposición las oficinas alrededor del mundo. El software de TA se puede integrar en los sistemas del *groupware* entregando a las organizaciones traducciones de bajo costo e inmediatas a cada empleado.

CONTENIDO MULTILINGÜE DE SITIOS EN INTERNET

Hoy, menos del 54 por ciento de los usuarios del Internet son nativo-hablantes del inglés. En los seis años próximos, el Internet experimentará un aumento del 150 por ciento en uso entre los no hablantes del inglés con la mayoría de nuevos usuarios de Asia/Pacífico y de los países latinoamericanos.

Las organizaciones globales que ofrecen un sitio *Web*¹⁴⁹ en inglés solamente encontrarán rápidamente que no pueden alcanzar el 57 por ciento de los usuarios del Internet del mundo. Para el comercio electrónico, un sitio en inglés podría acarrear su quiebra. Navegando en Internet y visitando los sitios de los líderes globales de la fortuna revela que estos líderes globales entienden la importancia de la comunicación multilingüe y han gastado mucho tiempo y dinero para tener sus sitios *Web* traducidos.

Desafortunadamente, muchas organizaciones debido al costo de la traducción de un sitio *Web* no pueden hacerlo. Para un sitio de comercio electrónico, traducir el contenido que puede cambiar diariamente e inclusive cada hora no es factible. En un mundo ideal, cada organización localizaría sus sitios *Web* para cada mercado que quiera atacar. Pero esto no será posible si no tienen la facilidad de traducir sus sitios *Web*.

La traducción en tiempo real automática es una solución rentable para la traducción del contenido dinámico del sitio *Web*. La calidad de la traducción del borrador puede ser mejorada perceptiblemente desarrollando los diccionarios modificados para los requisitos particulares incluyendo la terminología específica a la industria y a la compañía. Las traducciones entregadas por un *software* de TA son ideales para las descripciones del producto encontradas en catálogos, *FAQ's*¹⁵⁰, resultados de búsqueda y noticias.

¹⁴⁹ La Web o WWW (World Wide Web), en español: Telaraña mundial

¹⁵⁰ Frequently Asked Questions. En español: Preguntas más frecuentes

ACCESO MULTILINGÜE A LOS DOCUMENTOS Y ARCHIVOS

Cada segundo en cualquier organización global, los documentos se están distribuyendo alrededor del mundo vía *e-mail*. Al descargar el documento, muchos receptores encontrarán que, debido a la lengua en la que el documento está escrito, no podrán leerlo. Las bibliotecas centrales incluyen a menudo los documentos escritos en diversos idiomas que puedan necesitar los empleados que no hablan esos idiomas profundamente. Por ejemplo, la inhabilidad de leer el documento una vez abierto debido a su lengua. La herramienta de traducción de documentos de un software de TA en un Intranet de la compañía permitirá que un empleado someta un archivo a traducción y reciba la versión traducida en cuestión de segundos.

ANEXO C ASPECTOS CULTURALES

La UNESCO¹⁵¹ en Octubre de 2003 publicó un documento denominado *“Recomendación Sobre la Promoción y el Uso del Plurilingüismo y el Acceso Universal al Ciberespacio”*¹⁵², donde plantea el punto de vista de los Estados Miembro acerca de lo importante que es para la sociedad mantener y promover la diversidad lingüística, así como lo es el facilitar el acceso a las redes de información mundiales. A continuación se plasma una percepción positiva de las recomendaciones dadas por la UNESCO, que si se tienen en cuenta por los Estados Miembro, daría un gran apoyo a la TA:

- El recomendar acuerdos internacionales que posibiliten la libre expresión hace parte de la constitución de la UNESCO y esta facilidad se puede lograr con la aparición de las nuevas tecnologías de la información y la comunicación.
- Hay actualmente un gran interés en la UNESCO por discutir sobre la diversidad de idiomas y el acceso universal a la información en las redes mundiales de información.
- Los Estados Miembros de la UNESCO están interesados en que se promueva la creación y digitalización de contenidos en muchos aspectos, que derriben la barrera de expresión reduciendo los obstáculos lingüísticos y que se adopten las medidas necesarias para que se facilite el acceso a todo el mundo a estos medios.
- La diversidad de idiomas es de gran importancia en la sociedad, así, es necesario que se fomente la enseñanza de los idiomas como una estrategia que permita la conservación de los mismos. Se recomienda entonces por parte de la UNESCO que se amplíe el apoyo a los países en desarrollo para la creación de productos electrónicos que faciliten esta enseñanza.
- La UNESCO habla sobre la importancia de crear grupos de investigación y desarrollo para la creación de sistemas de búsqueda y herramientas de traducción que puedan ser usados por todo el mundo como soporte para acceder a la mayor cantidad de información que les sea posible.

¹⁵¹ United Nations Educational, Scientific and Cultural Organization. En español: Organización de las Naciones Unidas para la Educación la Ciencia y la Cultura

¹⁵² UNESCO, Proyecto de recomendación sobre la promoción y el uso del plurilingüismo y el acceso universal al ciberespacio e informe del director general. 2001.

- La UNESCO tiene interés en que haya un observatorio internacional que se encargue de hacer control de las políticas y normas que actúen sobre los recursos y aplicaciones plurilingües.
- Hay un gran interés en la UNESCO para que los países miembros y organismos internacionales se esfuercen en promover el acceso a Internet con mayores facilidades en especial para organizaciones públicas y sectores menos favorecidos. De igual manera, se tiene interés en que se facilite la *alfabetización electrónica*, así, se estaría logrando una mayor cobertura de este servicio y como consecuencia de esto habría mayor intercambio de información a nivel mundial.
- La UNESCO hace un llamado a todos los países para que pongan a disposición de todo el mundo los depósitos de información y demás archivos públicos, permitiendo la participación ciudadana en los estados democráticos. Como aporte, plantean la idea de formar un cuerpo de conocimiento de acceso universal, con la información generada por los proyectos y programas de desarrollo.
- En el aspecto legal, la UNESCO tiene la intención de formar un grupo con diferentes organizaciones gubernamentales del mundo que se encargue de recopilar información legislativa sobre la publicación de información de dominio público.
- La UNESCO tiene presente que hay una barrera en cuanto el acceso libre a información, llamada derechos de autor. Hay que tener en cuenta este aspecto y para ello recomienda que los Estados Miembros sean intermediarios con los propietarios de estos derechos para que en lo posible permitan la publicación universal de sus contenidos.
- Por último, se plantea el compromiso que deben tener los Estados Miembros para que estas disposiciones se divulguen a nivel mundial en los organismos correspondientes para que se apliquen estas recomendaciones.

ANEXO D DATOS ESTADISTICOS

PARES DE IDIOMAS

Existen 160 diferentes combinaciones de idiomas para las que existen herramientas de traducción, se destacan el inglés con 37 pares de idiomas, el alemán con 22 y el japonés con 12. el español posee 9 combinaciones (inglés, francés, alemán, japonés, italiano, catalán, ruso, coreano y portugués).

<i>Albanian--->German</i>	<i>English--->Hindi</i>	<i>French--->Greek</i>
<i>Arabic--->English</i>	<i>English--->Hungarian</i>	<i>French--->Italian</i>
<i>Arabic--->French</i>	<i>English--->Indonesian</i>	<i>French--->Japanese</i>
<i>Bosnian--->English</i>	<i>English--->Italian</i>	<i>French--->Korean</i>
<i>Catalan--->English</i>	<i>English--->Japanese</i>	<i>French--->Portuguese</i>
<i>Catalan--->Spanish</i>	<i>English--->Korean</i>	<i>French--->Russian</i>
<i>Chinese--->English</i>	<i>English--->Malay</i>	<i>French--->Spanish</i>
<i>Chinese--->Japanese</i>	<i>English--->Norwegian</i>	<i>German--->Albanian</i>
<i>Chinese--->Korean</i>	<i>English--->Papiamentu</i>	<i>German--->Croatian</i>
<i>Croatian--->English</i>	<i>English--->Pashto</i>	<i>German--->Czech</i>
<i>Croatian--->German</i>	<i>English--->Persian see</i>	<i>German--->Danish</i>
<i>Czech--->English</i>	<i>English--->Dari</i>	<i>German--->Dutch</i>
<i>Czech--->German</i>	<i>English--->Polish</i>	<i>German--->English</i>
<i>Danish--->German</i>	<i>English--->Portuguese</i>	<i>German--->Finnish</i>
<i>Dutch--->English</i>	<i>English--->Russian</i>	<i>German--->French</i>
<i>Dutch--->French</i>	<i>English--->Serbian</i>	<i>German--->Hungarian</i>
<i>Dutch--->German</i>	<i>English--->Slovak</i>	<i>German--->Italian</i>
<i>English--->Arabic</i>	<i>English--->Spanish</i>	<i>German--->Japanese</i>
<i>English--->Bosnian</i>	<i>English--->Thai</i>	<i>German--->Korean</i>
<i>English--->Bulgarian</i>	<i>English--->Turkish</i>	<i>German--->Latin</i>
<i>English--->Catalan</i>	<i>English--->Ukrainian</i>	<i>German--->Low</i>
<i>English--->Chinese</i>	<i>English--->Vietnamese</i>	<i>German</i>
<i>English--->Croatian</i>	<i>Esperanto--->English</i>	<i>German--->Polish</i>
<i>English--->Czech</i>	<i>Esperanto---</i>	<i>German---</i>
<i>English--->Dari</i>	<i>>Japanese</i>	<i>>Portuguese</i>
<i>English--->Dutch</i>	<i>Farsi--->English</i>	<i>German--->Romanian</i>
<i>English--->Esperanto</i>	<i>Finnish--->English</i>	<i>German--->Russian</i>
<i>English--->Farsi</i>	<i>Finnish--->German</i>	<i>German--->Slovak</i>
<i>English--->French</i>	<i>French--->Arabic</i>	<i>German--->Spanish</i>
<i>English--->German</i>	<i>French--->Dutch</i>	<i>German--->Swedish</i>
<i>English--->Greek</i>	<i>French--->English</i>	<i>German--->Tagalog</i>
<i>English--->Hebrew</i>	<i>French--->German</i>	<i>Greek--->English</i>

Greek--->French
 Hebrew--->English
 Hungarian--->English
 Hungarian--->German
 Indonesian--->English
 Italian--->English
 Italian--->French
 Italian--->German
 Italian--->Japanese
 Italian--->Korean
 Italian--->Portuguese
 Italian--->Russian
 Italian--->Spanish
 Japanese--->Chinese
 Japanese--->English
 Japanese--->French
 Japanese--->German
 Japanese--->Italian
 Japanese--->Korean
 Japanese--->Malay
 Japanese--->Polish
 Japanese---
 >Portuguese
 Japanese--->Russian
 Japanese--->Spanish

Japanese--->Ukrainian
 Korean--->Chinese
 Korean--->English
 Korean--->Japanese
 Latin--->German
 Low German---
 >German
 Malay--->English
 Papiamentu--->English
 Polish--->English
 Polish--->German
 Polish--->Japanese
 Portuguese--->English
 Portuguese--->French
 Portuguese---
 >German
 Portuguese--->Italian
 Portuguese---
 >Japanese
 Portuguese--->Korean
 Portuguese---
 >Spanish
 Romanian--->German
 Russian--->English
 Russian--->French

Russian--->German
 Russian--->Italian
 Russian--->Japanese
 Russian--->Spanish
 Russian--->Ukrainian
 Serbian--->English
 Slovak--->English
 Slovak--->German
 Spanish--->Catalan
 Spanish--->English
 Spanish--->French
 Spanish--->German
 Spanish--->Italian
 Spanish--->Japanese
 Spanish--->Korean
 Spanish---
 >Portuguese
 Spanish--->Russian
 Swedish--->English
 Swedish--->German
 Tagalog--->German
 Ukrainian--->English
 Ukrainian--->Japanese
 Ukrainian--->Russian

COMBINACIONES PARA LAS QUE SE CUENTA CON DICCIONARIO ELECTRÓNICO

Existen 146 combinaciones para las cuales se cuenta con un diccionario electrónico. Es importante destacar que en realidad existen muchas más combinaciones, ya que para idiomas en los cuales existen diccionarios multilingües el compendio en la mayoría de ocasiones no especifica cuales o cuantos son los idiomas que maneja. De estas 17 combinaciones involucran al idioma español.

Afrikaans--->many*
 Afrikaans--->English
 Albanian--->many*
 Albanian--->English
 Arabic--->many*
 Arabic--->English

Basque--->many*
 Bosnian--->many*
 Bulgarian--->many*
 Bulgarian--->English
 Catalan--->many*
 Chinese--->many*

Chinese--->English
 Chinese--->Japanese
 Croatian--->many*
 Croatian--->English
 Czech--->many*
 Czech--->English

Czech--->German
 Danish--->many*
 Danish--->English
 Danish--->Finnish
 Dutch--->many*
 Dutch--->English
 Dutch--->German
 English--->many*
 English--->Arabic
 English--->Chinese
 English--->French
 English--->German
 English--->Hungarian
 English--->Japanese
 English--->Latin
 English--->Portuguese
 English--->Romanian
 English--->Russian
 English--->Spanish
 Esperanto--->many*
 Faroese--->many*
 Farsi--->many*
 Farsi--->English
 Finnish--->many*
 Finnish--->English
 Flemish--->many*
 French--->many*
 French--->English
 French--->Finnish
 French--->German
 French--->Hungarian
 French--->Italian
 French--->Polish
 French--->Russian
 French--->Spanish
 Frisian--->many*
 Gaelic--->many*
 German--->many*
 German--->Dutch
 German--->English
 German--->Finnish
 German--->French
 German--->Hungarian

German--->Italian
 German--->Latin
 German--->Russian
 German--->Spanish
 Greek--->many*
 Greek--->English
 Hebrew--->many*
 Hebrew--->English
 Hebrew--->Russian
 Hindi--->many*
 Hindi--->English
 Hungarian--->many*
 Hungarian--->English
 Hungarian--->German
 Icelandic--->many*
 Indonesian--->many*
 Indonesian--->English
 Italian--->many*
 Italian--->English
 Italian--->Finnish
 Italian--->French
 Italian--->German
 Italian--->Hungarian
 Italian--->Russian
 Japanese--->many*
 Japanese--->Chinese
 Japanese--->English
 Korean--->many*
 Korean--->English
 Latin--->many*
 Latin--->English
 Latin--->German
 Latvian--->English
 Malay--->many*
 Norwegian--->many*
 Norwegian--->English
 Papiamentu--->many*
 Polish--->many*
 Polish--->English
 Portuguese--->many*
 Portuguese--->English
 Portuguese--->Finnish
 Romanian--->many*

Romanian--->English
 Russian--->many*
 Russian--->English
 Russian--->Finnish
 Russian--->French
 Russian--->German
 Russian--->Hungarian
 Russian--->Italian
 Russian--->Spanish
 Russian--->Swedish
 Serbian--->many*
 Serbian--->English
 Slovak--->many*
 Slovenian--->many*
 Slovenian--->English
 Spanish--->many*
 Spanish--->English
 Spanish--->Finnish
 Spanish--->French
 Spanish--->German
 Spanish--->Russian
 Sranan--->many*
 Swahili--->many*
 Swahili--->English
 Swedish--->many*
 Swedish--->English
 Swedish--->Finnish
 Swedish--->German
 Swedish--->Russian
 Tagalog--->many*
 Thai--->many*
 Turkish--->many*
 Turkish--->English
 Ukrainian--->many*
 Ukrainian--->English
 Vietnamese--->many*
 Vietnamese--->English
 Welsh--->many*
 Yiddish--->many*
 Yiddish--->English
 Yucatan Maya--->many*
 Zulu--->many*

COMBINACIONES QUE INVOLUCRAN AL ESPAÑOL

Catalan--->Spanish
English--->Spanish
French--->Spanish
German--->Spanish
Italian--->Spanish
Japanese--->Spanish
Portuguese--->Spanish
Russian--->Spanish
Spanish--->Catalan
Spanish--->English
Spanish--->French
Spanish--->German
Spanish--->Italian
Spanish--->Japanese
Spanish--->Korean
Spanish--->Portuguese
Spanish--->Russian

CANTIDAD DE EMPRESAS EN EL MERCADO

La lista de Compañías listadas en el compendio se incluye productores, vendedores, distribuidores y proveedores de servicios de sistemas de TA. Se listan un total de 198 empresas conocidas:

A.I. Soft, Inc.
ABBYY Software House
ALIS Technologies
ASCII Solutions
ATA Software Technology Ltd
ATA Software Technology Ltd.
ATIA Ltd.
Aaron David Irvine, Queen's
University Belfast
Ahead Software AG
Alchemy Software Development Ltd.
AlphaSoft
Altavista
Amikai Inc.
AppTek

AppTek Inc
AppTek Inc.
Applied Information Technologies AG
Aquino Software
ArabNet Technology
Arsenal Inc.
Atril Software
Automatictrans S.L.
Avral Technologies Ltd.
Babel Informática
Babylon Ltd.
Basis Technology Corporation
Beijing ShiDa Ming Tai Computer
Application Technology Co.Ltd.

Beijing YaXinCheng Software
 Technology Co.Ltd.
 Bencom Digital Inc.
 Bilingual Software
 Bilsag Ltd.
 Bowne Global Solutions
 Brall Software
 BridgeTerm
 Brother Industries Ltd
 Caissa International Corporation
 Centre for Development of Advanced
 Computing
 Champollion WordFast Ltd.
 ChangshinSoft
 Cimos
 Ciyasoft Corporation
 ClickQ Co.Ltd.
 ComCul International
 Compendium Deutschland GmbH
 Convey Software
 Create Osaka Co.
 Cross Language Inc.
 D'Agostini Organizzazione
 Delta Translator
 Digiko Soft
 Dream C&C
 ESteam Ltd.
 EWGate Pte Ltd.
 Ectaco
 Ectaco Inc.
 Evolutionary Solutions
 Fujitsu Ltd.
 Gakken Co.Ltd.
 GalTech Soft
 Gerard van Wilgen
 GlobalSight
 GlobalWare AG
 Google
 Heartsome
 Helicon Software Development
 Holon Inc.
 Holtschke GmbH
 Hostran & Microc Software

Huajian Group (China)
 Humanitas-International
 IBM Corporation
 IBM Japan
 ITC (Innovative Technology Center)
 Inc.
 ITR International Translation
 Resources Ltd.
 IdiomaX
 Idiomax
 Ifec S.R.O.
 Impulse Japan Inc.
 Indian Institute of Technology, Kanpur
 InstallShield Software Corporation
 Institute for Language and Speech
 Processing
 International Science and Technology
 Associates, Inc.
 Inventec Corporation
 Japan Science and Technology
 Corporation (JST)
 Jin Shan Computer Software
 Company
 John Chandioux Consultants Inc.,
 EDIT Inc.
 Julia Emily Software
 Justsystem Corp.
 Justsystem Corporation
 Kevin Solway
 Kielikone Ltd.
 Kingsoft
 Kodensha
 Kodensha Co.Ltd.
 Kornyei Tibor
 LNI Soft
 LangSoft
 Langsoft
 Language Dynamics Corporation
 Language Engineering Corporation.
 Language Technology Centre
 Language Weaver Inc.
 Lexicool.com
 Linguattec Sprachtechnologien

<i>Linguattech International</i>	<i>SDL International.</i>
<i>Lingvistica '98 Inc.</i>	<i>SDL international</i>
<i>Lingvistica '98 Inc. & Lingvistica b.v.</i>	<i>SKIK spol s.r.o.</i>
<i>Lingvistica'98 Inc.</i>	<i>STAR AG</i>
<i>LogoMedia</i>	<i>SYNTHEMA Srl</i>
<i>LogoMedia Corporation</i>	<i>Sakhr Software</i>
<i>LogoVista Corporation</i>	<i>Sakhr Software Co.</i>
<i>Lomac</i>	<i>Samlight International Network Ltd.</i>
<i>Lotus Development Corporation</i>	<i>Schaudin</i>
<i>Medialingua</i>	<i>Seagrand Co.Ltd.</i>
<i>MetaTaxis Software and Services</i>	<i>Seiko Instruments</i>
<i>Microton</i>	<i>Sharp Corporation</i>
<i>Morphologic Ltd.</i>	<i>SkyCode Ltd.</i>
<i>MultiCorpora Inc.</i>	<i>Smart Communications Inc.</i>
<i>Mutilizer Inc.</i>	<i>Smart Link Corporation</i>
<i>NEC</i>	<i>Socatra</i>
<i>NEC Corporation</i>	<i>Softissimo</i>
<i>NTT MSC Sdn.Bhd.</i>	<i>Softissimo Ltd.</i>
<i>NanaTech</i>	<i>Software Technology Co.Ltd.</i>
<i>National Center for Technological</i>	<i>SourceForge.net</i>
<i>Progress (Vietnam)</i>	<i>SourceForge.net (Open Source</i>
<i>Oki Electric Co.Ltd.</i>	<i>Development Network)</i>
<i>Omron Software</i>	<i>Sourcenext</i>
<i>Otek International Inc.</i>	<i>Stargate A/S</i>
<i>Otek International Inc./TransWhiz</i>	<i>Systran Software Inc.</i>
<i>Software Inc.</i>	<i>TEMIS</i>
<i>P.H.Brink</i>	<i>Targumatik</i>
<i>PAC do Brasil Sistemas para</i>	<i>Telelingua International</i>
<i>Computador e Comércio Ltda.</i>	<i>Terminotix Ltd.</i>
<i>PASS Engineering GmbH</i>	<i>Terra Lycos</i>
<i>Padideh Co.</i>	<i>Toggletext</i>
<i>Pan American Health Organization</i>	<i>Toshiba Solutions</i>
<i>Pan American Health Organization</i>	<i>Trados</i>
<i>Paragon Software.</i>	<i>Translation Experts Ltd.</i>
<i>PetaMem GmbH</i>	<i>TranslationWave.com</i>
<i>PhatWare Corporation</i>	<i>Translations.com</i>
<i>Powerglot Software</i>	<i>Transparent Language</i>
<i>ProLangs Ltd.</i>	<i>Transtar Company, System Nihon</i>
<i>Project MT Ltd.</i>	<i>Science Co.Ltd.</i>
<i>ProjectMT Ltd.</i>	<i>Tranzsend</i>
<i>Prolingua</i>	<i>Trident Software</i>
<i>Quick-Pen.com</i>	<i>Trilingual Peksong</i>
<i>SDL International</i>	<i>Ultralingua Software</i>

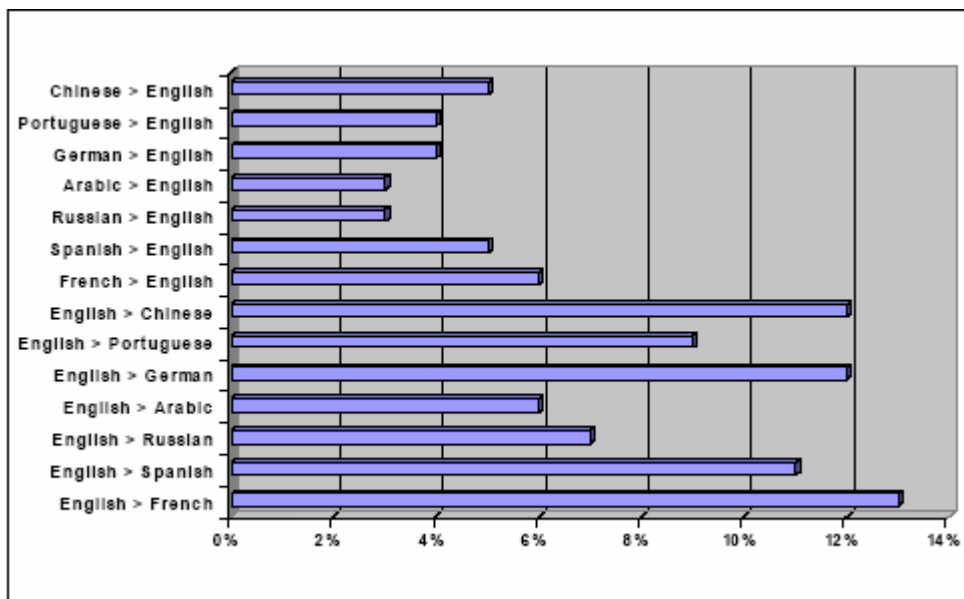
UniSite Software
Unikotech
Unisoft
Universitat d'Alacant
Universitat d'Alacant, Departament de
Llenguages i Sistemes Informàtics
VTT Information Technology

VoxTec LLC
WizArt
WordBank Co.Ltd.
WordMagic
Worldlingo Inc.
Xplanation
classicalmusic.gr

DEMANDA MUNDIAL

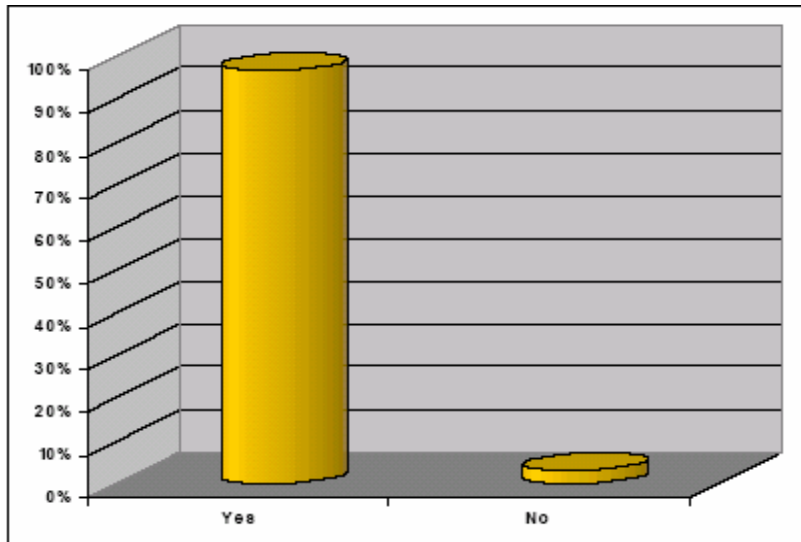
Las siguientes estadísticas son resultado de un estudio realizado por el Banco Mundial en Agosto de 2004, basándose en organizaciones internacionales gubernamentales, organizaciones no gubernamentales, compañías de traducción, localización y otras.

COMBINACIONES DE IDIOMAS PARA LAS QUE SE USA TRADUCTORES INTERNOS



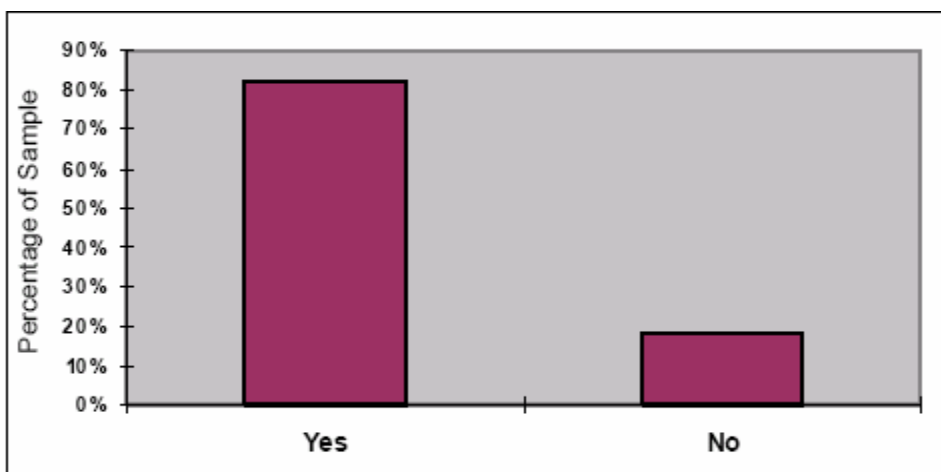
Se puede ver que el inglés es el idioma más demandado como fuente y como destino.

MONTO DE EMPRESAS QUE SUBCONTRATAN SERVICIOS DE TRADUCCIÓN PARA COMBINACIONES DE IDIOMAS EN LAS QUE NO SE TIENE PERSONAL INTERNO



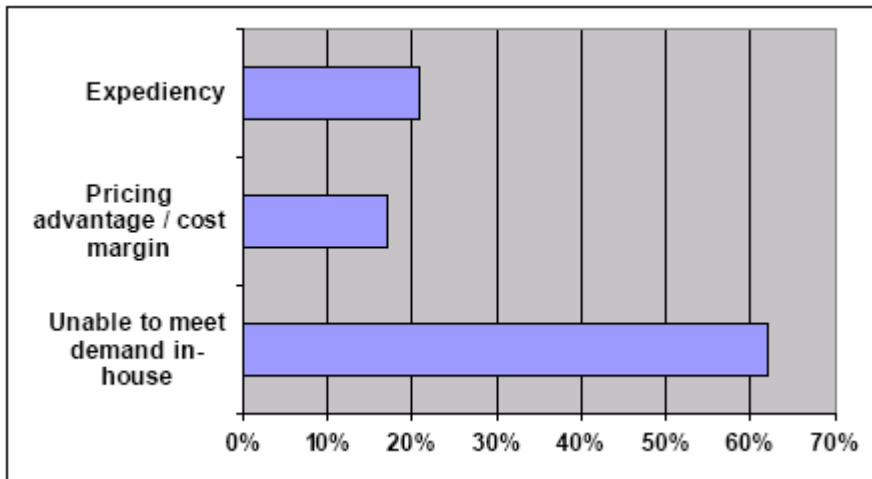
Casi el 100% de las empresas que requieren de información en otro idioma, contratan un *outsourcing* de traducción.

MONTO DE EMPRESAS QUE SUBCONTRATAN SERVICIOS DE TRADUCCIÓN PARA COMBINACIONES DE IDIOMAS EN LAS QUE SI SE TIENE PERSONAL INTERNO



Incluso las empresas que necesitan de información en otro idioma y que tienen traductores para estos idiomas, contratan traductores para realizar estas labores.

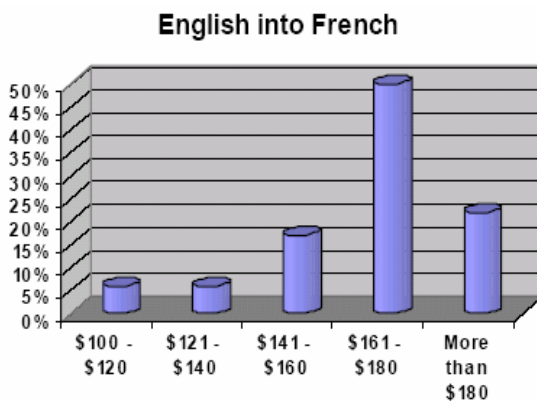
RAZONES POR LAS QUE SE SUBCONTRATA ESTE SERVICIO



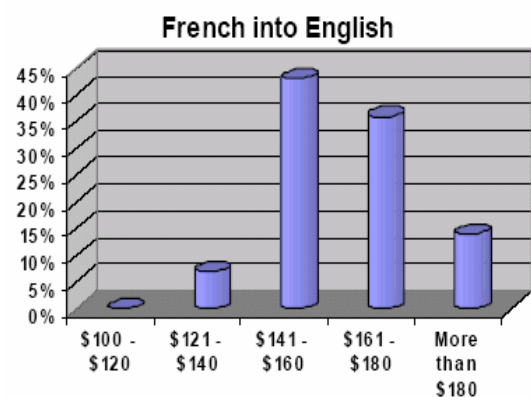
La principal razón para contratar los servicios de traducción, así se cuente con traductores competentes, es la incapacidad de suplir la demanda.

MONTOS DE DINERO QUE SE PAGA A LAS EMPRESAS QUE PRESTAN SERVICIOS DE TRADUCCIÓN POR CADA 1000 PALABRAS

Inglés-Francés



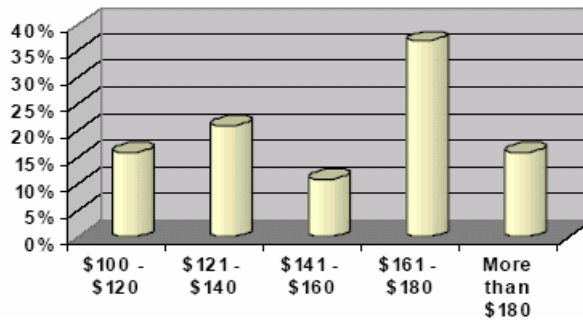
Francés-Inglés



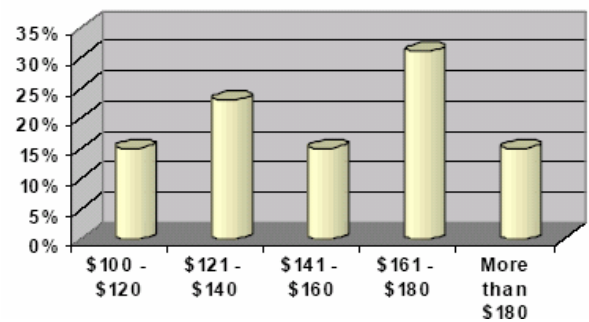
Inglés-Español

Español-Inglés

English into Spanish

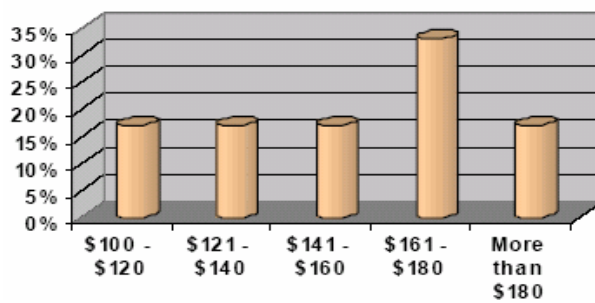


Spanish into English



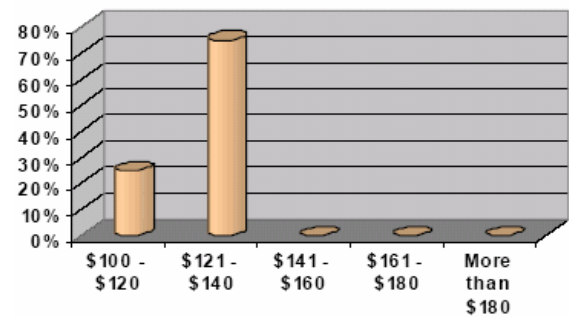
Inglés-Ruso

English into Russian



Ruso-Inglés

Russian into English



ANEXO E PRHASE¹⁵³

In grammar, a phrase is a group of words that functions as a single unit in the syntax of a sentence.

For example the house at the end of the street (example 1) is a phrase. It acts like a noun. It contains the phrase at the end of the street (example 2), which acts like an adjective. Example 2 could be replaced by white, to make the phrase the white house. Examples 1 and 2 contain the phrase the end of the street (example 3) which acts like a noun. It could be replaced by the cross-roads to give the house at the cross-roads.

Each phrase has a word called its head which links it to the rest of the sentence. In English the head is often the first word of the phrase.

Phrases may be classified by the type of head they take

- Prepositional phrase (PP) with a preposition as head (e.g. in love, over the rainbow). Languages that use postpositions instead have postpositional phrases. The two types are sometimes commonly referred to as adpositional phrases.
- Noun phrase (NP) with a noun as head (e.g. the black cat, a cat on the mat)
- Verb phrase (VP) with a verb as head (e.g. eat cheese, jump up and down)
- Adjectival phrase with an adjective as head (e.g. full of toys)
- Adverbial phrase with adverb as head (e.g. very carefully)

FORMAL DEFINITION

A phrase is a syntactic structure which has syntactic properties derived from its head.

For example the house at the end of the street is a noun phrase. Its head is house, and its syntactic properties come from that fact. It contains prepositional phrase at the end of the street, which acts as an adjunct. At the end of the street could be replaced by another adjunct, such as white, to make the phrase the white house. Of the street, another prepositional phrase, acts as a complement of end. Each phrase has a word called its head which gives it its syntactic properties.

¹⁵³ Tomado de Wikipedia. <<http://en.wikipedia.org/wiki/Phrase>>

COMPLEXITY

A complex phrase consists of several words, whereas a simple phrase consists of only one word. This terminology is especially often used with verb phrases:

- simple past and present are simple verb, which require just one verb
- complex verb have one or two aspects added, hence require additional two or three words

"Complex", which is phrase-level, is often confused with "compound", which is word-level. However, there are certain phenomena that formally seem to be phrases but semantically are more like compounds, like "women's magazines", which has the form of a possessive noun phrase, but which refers (just like a compound) to one specific lexeme (i.e. a magazine for women and not some magazine owned by a woman).

SEMIOTIC APPROACHES TO THE CONCEPT OF "PHRASE"

In more semiotic approaches to language, such, as for instance, the more cognitivist versions of construction grammar, a phrasal structure is not only a certain formal combination of word types whose features are inherited from the head. Here each phrasal structure also expresses some type of conceptual content, be it specific or abstract.

For example prepositional phrases express a figure-ground relation in which the prepositional complement is the ground, the preposition itself specifies the relation, and the precedent element is the figure.

Thus, in semiotic approaches to phrasal structure, a phrase not only has a specific formal configuration, but is also characterized by a recognizable (abstract or specific) semantic content.

PHRASE STRUCTURE RULES

Phrase-structure rules were used in early transformational grammar (TGG) to describe a given language's syntax. They were used to break a natural language sentence down into its constituent parts (also known as syntactic categories) namely phrasal categories and lexical categories (aka parts of speech). Phrasal categories include the noun phrase, verb phrase, and prepositional phrase; lexical categories include noun, verb, adjective, adverb, and many others. Phrase structure rules were not an invention of TGG; rather, early TGG's defining characteristics were those systems which it had in addition to phrase structure

rules. A grammar which uses phrase structure rules is called a phrase structure grammar.

Phrase structure rules are usually of the form $A \rightarrow B \ C$, meaning that the constituent A is separated into the two subconstituents B and C . Some examples are:

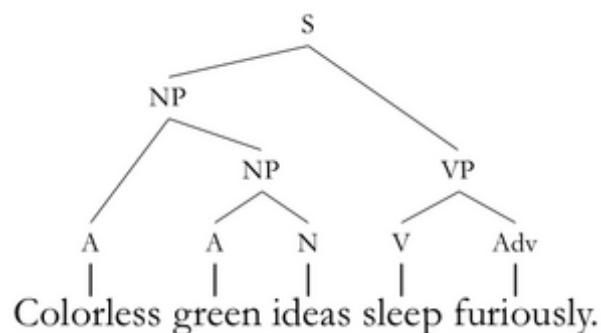
$$\begin{aligned} S &\rightarrow NP \ VP \\ NP &\rightarrow Det \ N1 \\ N1 &\rightarrow (AP) \ N1 \ (PP) \end{aligned}$$

The first rule reads: An S consists of an NP followed by a VP . This means A sentence consists of a noun phrase followed by a verb phrase. The next one: A noun phrase consists of a determiner followed by a noun.

Further explanations of the constituents: S , Det , NP , VP , AP

Associated with phrase structure rules is a famous example of a grammatically correct sentence. The sentence was constructed by Noam Chomsky as an illustration that syntactically but not semantically correct sentences are possible.

Colorless green ideas sleep furiously can be diagrammed as a **phrase tree**, as below: Where S represents a grammatical sentence.



ANEXO F THE BROWN CORPUS TAG-SET

Listed alphabetically below are the complete set of tags used in the Brown corpus. Each tag has examples of the tokens that were annotated with that tag. The examples are taken directly from the Brown corpus.

Unlike some of the other tag-sets handled by the AMALGAM tagger Brown uses 'combined tags' for word such as **won't** (MD*) and **I'd** (PPSS+HVD). Combined tags come in only these two forms. Either negated words have an asterisk appended after their tag or the plus symbol separates the tags for the different tokens that make up the complete combined word. This makes it a trivial task to split combined tags. AMALGAM's version of the Brown tagger annotates with combined tags only if the tokenizer is switched off. This is done on the e-mail server by appending "notoken" to the start of the subject line of the e-mail message. If the tokenizer is used the combined words are split into their constituent parts and the tags applied to each part. So, **won't** (MD*) becomes **will** (MD) plus **n't** (*) and **I'd** (PPSS+HVD) becomes **I** (PPSS) plus **'d** (HVD).

Further information on the Brown corpus can be found at the International Computer Archive of Modern English (ICAME) corpus collection. Automatic grammatical tagging of English. Providence, R.I.: Department of Linguistics, Brown University.

Tag	Description	Examples
(opening parenthesis	(
)	closing parenthesis)
*	negator	not n't
,	comma	,
--	dash	--
.	sentence terminator	. ? ; ! :
:	colon	:
ABL	determiner/pronoun, pre-qualifier	quite such rather
ABN	determiner/pronoun, pre-quantifier	all half many nary

ABX	determiner/pronoun, double conjunction or pre-quantifier	both
AP	determiner/pronoun, post-determiner	many other next more last former little several enough most least only very few fewer past same Last latter less single plenty 'nough lesser certain various manye next-to-last particular final previous present nuf
AP\$	determiner/pronoun, post-determiner, genitive	other's
AP+AP	determiner/pronoun, post-determiner, hyphenated pair	many-much
AT	article	the an no a every th' ever' ye
BE	verb "to be", infinitive or imperative	be
BED	verb "to be", past tense, 2nd person singular or all persons plural	were
BED*	verb "to be", past tense, 2nd person singular or all persons plural, negated	weren't
BEDZ	verb "to be", past tense, 1st and 3rd person singular	was
BEDZ*	verb "to be", past tense, 1st and 3rd person singular, negated	wasn't
BEG	verb "to be", present participle or gerund	being
BEM	verb "to be", present tense, 1st person singular	am
BEM*	verb "to be", present tense, 1st person singular, negated	ain't
BEN	verb "to be", past participle	been
BER	verb "to be", present tense, 2nd person singular or all persons plural	are art
BER*	verb "to be", present tense, 2nd person singular or all persons plural, negated	aren't ain't
BEZ	verb "to be", present tense, 3rd person singular	is
BEZ*	verb "to be", present tense, 3rd person singular, negated	isn't ain't
CC	conjunction, coordinating	and or but plus & either neither nor yet 'n' and/or minus an'
CD	numeral, cardinal	two one 1 four 2 1913 71 74 637 1937 8 five three million 87-31 29-5 seven 1,119 fifty-three 7.5 billion hundred 125,000 1,700 60 100 six ...
CD\$	numeral, cardinal, genitive	1960's 1961's .404's

CS	conjunction, subordinating	that as after whether before while like because if since for than altho until so unless though providing once lest s'posin' till whereas whereupon supposing tho' albeit then so's 'fore
DO	verb "to do", uninflected present tense, infinitive or imperative	do dost
DO*	verb "to do", uninflected present tense or imperative, negated	don't
DO+PPSS	verb "to do", past or present tense + pronoun, personal, nominative, not 3rd person singular	d'you
DOD	verb "to do", past tense	did done
DOD*	verb "to do", past tense, negated	didn't
DOZ	verb "to do", present tense, 3rd person singular	does
DOZ*	verb "to do", present tense, 3rd person singular, negated	doesn't don't
DT	determiner/pronoun, singular	this each another that 'nother
DT\$	determiner/pronoun, singular, genitive	another's
DT+BEZ	determiner/pronoun + verb "to be", present tense, 3rd person singular	that's
DT+MD	determiner/pronoun + modal auxiliary	that'll this'll
DTI	determiner/pronoun, singular or plural	any some
DTS	determiner/pronoun, plural	these those them
DTS+BEZ	pronoun, plural + verb "to be", present tense, 3rd person singular	them's
DTX	determiner, pronoun or double conjunction	neither either one
EX	existential there	there
EX+BEZ	existential there + verb "to be", present tense, 3rd person singular	there's
EX+HVD	existential there + verb "to have", past tense	there'd
EX+HVZ	existential there + verb "to have", present tense, 3rd person singular	there's
EX+MD	existential there + modal auxiliary	there'll there'd
FW-*	foreign word: negator	pas non ne

FW-AT	foreign word: article	la le el un die der ein keine eine das las les ll
FW-AT+NN	foreign word: article + noun, singular, common	l'orchestre l'identite l'arcade l'ange l'assistance l'activite L'Universite l'independance L'Union L'Unita l'osservatore
FW-AT+NP	foreign word: article + noun, singular, proper	L'Astree L'Imperiale
FW-BE	foreign word: verb "to be", infinitive or imperative	sit
FW-BER	foreign word: verb "to be", present tense, 2nd person singular or all persons plural	sind sunt etes
FW-BEZ	foreign word: verb "to be", present tense, 3rd person singular	ist est
FW-CC	foreign word: conjunction, coordinating	et ma mais und aber och nec y
FW-CD	foreign word: numeral, cardinal	une cinq deux sieben unam zwei
FW-CS	foreign word: conjunction, subordinating	bevor quam ma
FW-DT	foreign word: determiner/pronoun, singular	hoc
FW-DT+BEZ	foreign word: determiner + verb "to be", present tense, 3rd person singular	c'est
FW-DTS	foreign word: determiner/pronoun, plural	haec
FW-HV	foreign word: verb "to have", present tense, not 3rd person singular	habe
FW-IN	foreign word: preposition	ad de en a par con dans ex von auf super post sine sur sub avec per inter sans pour pendant in di
FW-IN+AT	foreign word: preposition + article	della des du aux zur d'un del dell'
FW-IN+NN	foreign word: preposition + noun, singular, common	d'etat d'hotel d'argent d'identite d'art
FW-IN+NP	foreign word: preposition + noun, singular, proper	d'Yquem d'Eiffel
FW-JJ	foreign word: adjective	avant Espagnol sinfonica Siciliana Philharmonique grand publique haute noire bouffe Douce meme humaine bel serieuses royaux anticus presto Sovietskaya Bayerische comique schwarzen ...
FW-JJR	foreign word: adjective, comparative	fortiori
FW-JJT	foreign word: adjective, superlative	optimo
FW-NN	foreign word: noun, singular, common	ballet esprit ersatz mano chatte goutte sang Fledermaus oud def kolkhoz roi troika canto boite blutwurst carne muzyka bonheur monde piece force ...

FW-NN\$	foreign word: noun, singular, common, genitive	corporis intellectus arte's dei aeternitatis senioritatis curiae patronne's chambre's
FW-NNS	foreign word: noun, plural, common	al culpas vopos boites hafliis kolkhozes augen tyrannis alpha-beta-gammas metis banditos rata phis negociants crus Einsatzkommandos kamikaze wohaws sabinas zorrillas palazzi engages coureurs corroborees yori Übermenschen ...
FW-NP	foreign word: noun, singular, proper	Karshilama Dieu Rundfunk Afrique Espanol Afrika Spagna Gott Carthago deus
FW-NPS	foreign word: noun, plural, proper	Svenskarna Atlantes Dieux
FW-NR	foreign word: noun, singular, adverbial	heute morgen aujourd'hui hoy
FW-OD	foreign word: numeral, ordinal	18e 17e quintus
FW-PN	foreign word: pronoun, nominal	Hoc
FW-PP\$	foreign word: determiner, possessive	Mea mon deras vos
FW-PPL	foreign word: pronoun, singular, reflexive	Se
FW-PPL+VBZ	foreign word: pronoun, singular, reflexive + verb, present tense, 3rd person singular	s'excuse s'accuse
FW-PPO	pronoun, personal, accusative	lui me moi mi
FW-PPO+IN	foreign word: pronoun, personal, accusative + preposition	mecum tecum
FW-PPS	foreign word: pronoun, personal, nominative, 3rd person singular	Il
FW-PPSS	foreign word: pronoun, personal, nominative, not 3rd person singular	ich vous sie je
FW-PPSS+HV	foreign word: pronoun, personal, nominative, not 3rd person singular + verb "to have", present tense, not 3rd person singular	j'ai
FW-QL	foreign word: qualifier	minus
FW-RB	foreign word: adverb	Bas assai deja um wiederum cito velociter vielleicht simpliciter non zu domi nuper sic forsan olim oui semper tout despues hors
FW-RB+CC	foreign word: adverb + conjunction, coordinating	forisque
FW-TO+VB	foreign word: infinitival to + verb, infinitive	d'entretenir
FW-UH	foreign word: interjection	sayonara bien adieu arigato bonjour adios bueno tchalo ciao o
FW-VB	foreign word: verb, present tense, not 3rd person singular, imperative or infinitive	Nolo contendere vive fermate faciunt esse vade noli tangere dites duces meminisse iuvabit gosaimasu voulez habla ksu'u'peli'afo lacheln miuchi say allons strafe portant

FW-VBD	foreign word: verb, past tense	stabat peccavi audivi
FW-VBG	foreign word: verb, present participle or gerund	nolens volens appellant seq. obliterans servanda dicendi delenda
FW-VBN	foreign word: verb, past participle	Vue verstrichen rasa verboten engages
FW-VBZ	foreign word: verb, present tense, 3rd person singular	gouverne sinkt sigue diapiace
FW-WDT	foreign word: WH-determiner	Quo qua quod que quok
FW-WPO	foreign word: WH-pronoun, accusative	quibusdam
FW-WPS	foreign word: WH-pronoun, nominative	Qui
HV	verb "to have", uninflected present tense, infinitive or imperative	have hast
HV*	verb "to have", uninflected present tense or imperative, negated	haven't ain't
HV+TO	verb "to have", uninflected present tense + infinitival to	hafta
HVD	verb "to have", past tense	Had
HVD*	verb "to have", past tense, negated	hadn't
HVG	verb "to have", present participle or gerund	having
HVN	verb "to have", past participle	Had
HVZ	verb "to have", present tense, 3rd person singular	has hath
HVZ*	verb "to have", present tense, 3rd person singular, negated	hasn't ain't
IN	preposition	of in for by considering to on among at through with under into regarding than since despite according per before toward against as after during including between without except upon out over ...
IN+IN	preposition, hyphenated pair	f'ovuh
IN+PPO	preposition + pronoun, personal, accusative	t'hi-im
JJ	adjective	recent over-all possible hard-fought favorable hard meager fit such widespread outmoded inadequate ambiguous grand clerical effective orderly federal foster general proportionate ...
JJ\$	adjective, genitive	Great's

JJ+JJ	adjective, hyphenated pair	big-large long-far
JJR	adjective, comparative	greater older further earlier later freer franker wider better deeper firmer tougher faster higher bigger worse younger lighter nicer slower happier frothier Greater newer Elder ...
JJR+CS	adjective + conjunction, coordinating	lighter'n
JJS	adjective, semantically superlative	top chief principal northernmost master key head main tops utmost innermost foremost uppermost paramount topmost
JJT	adjective, superlative	best largest coolest calmest latest greatest earliest simplest strongest newest fiercest unhappiest worst youngest worthiest fastest hottest fittest lowest finest smallest staunchest ...
MD	modal auxillary	should may might will would must can could shall ought need wilt
MD*	modal auxillary, negated	cannot couldn't wouldn't can't won't shouldn't shan't mustn't musn't
MD+HV	modal auxillary + verb "to have", uninflected form	shouldda musta coulda must've woulda could've
MD+PPSS	modal auxillary + pronoun, personal, nominative, not 3rd person singular	willya
MD+TO	modal auxillary + infinitival to	oughta
NN	noun, singular, common	failure burden court fire appointment awarding compensation Mayor interim committee fact effect airport management surveillance jail doctor intern extern night weekend duty legislation Tax Office ...
NN\$	noun, singular, common, genitive	season's world's player's night's chapter's golf's football's baseball's club's U.'s coach's bride's bridegroom's board's county's firm's company's superintendent's mob's Navy's ...
NN+BEZ	noun, singular, common + verb "to be", present tense, 3rd person singular	water's camera's sky's kid's Pa's heat's throat's father's money's undersecretary's granite's level's wife's fat's Knife's fire's name's hell's leg's sun's roulette's cane's guy's kind's baseball's ...
NN+HVD	noun, singular, common + verb "to have", past tense	Pa'd
NN+HVZ	noun, singular, common + verb "to have", present tense, 3rd person singular	guy's Knife's boat's summer's rain's company's
NN+IN	noun, singular, common + preposition	buncha
NN+MD	noun, singular, common + modal auxillary	cowhand'd sun'll
NN+NN	noun, singular, common, hyphenated pair	stomach-belly

NNS	noun, plural, common	irregularities presentments thanks reports voters laws legislators years areas adjustments chambers \$100 bonds courts sales details raises sessions members congressmen votes polls calls ...
NNS\$	noun, plural, common, genitive	taxpayers' children's members' States' women's cutters' motorists' steelmakers' hours' Nations' lawyers' prisoners' architects' tourists' Employers' secretaries' Rogues' ...
NNS+MD	noun, plural, common + modal auxillary	duds'd oystchers'll
NP	noun, singular, proper	Fulton Atlanta September-October Durwood Pye Ivan Allen Jr. Jan. Alpharetta Grady William B. Hartsfield Pearl Williams Aug. Berry J. M. Cheshire Griffin Opelika Ala. E. Pelham Snodgrass ...
NP\$	noun, singular, proper, genitive	Green's Landis' Smith's Carreon's Allison's Boston's Spahn's Willie's Mickey's Milwaukee's Mays' Howsam's Mantle's Shaw's Wagner's Rickey's Shea's Palmer's Arnold's Broglio's ...
NP+BEZ	noun, singular, proper + verb "to be", present tense, 3rd person singular	W.'s Ike's Mack's Jack's Kate's Katharine's Black's Arthur's Seaton's Buckhorn's Breed's Penny's Rob's Kitty's Blackwell's Myra's Wally's Lucille's Springfield's Arlene's
NP+HVZ	noun, singular, proper + verb "to have", present tense, 3rd person singular	Bill's Guardino's Celie's Skolman's Crosson's Tim's Wally's
NP+MD	noun, singular, proper + modal auxillary	Gyp'll John'll
NPS	noun, plural, proper	Chases Aderholds Chapelles Armisteads Lockies Carbones French Marskmen Toppers Franciscans Romans Cadillacs Masons Blacks Catholics British Dixiecrats Mississippians Congresses ...
NPS\$	noun, plural, proper, genitive	Republicans' Orioles' Birds' Yanks' Redbirds' Bucs' Yankees' Stevenses' Geraghtys' Burkes' Wackers' Achaeans' Dresbachs' Russians' Democrats' Gershwins' Adventists' Negroes' Catholics' ...
NR	noun, singular, adverbial	Friday home Wednesday Tuesday Monday Sunday Thursday yesterday tomorrow tonight West East Saturday west left east downtown north northeast southeast northwest North South right ...
NR\$	noun, singular, adverbial, genitive	Saturday's Monday's yesterday's tonight's tomorrow's Sunday's Wednesday's Friday's today's Tuesday's West's Today's South's
NR+MD	noun, singular, adverbial + modal auxillary	today'll
NRS	noun, plural, adverbial	Sundays Mondays Saturdays Wednesdays Souths Fridays

OD	numeral, ordinal	first 13th third nineteenth 2d 61st second sixth eighth ninth twenty-first eleventh 50th eighteenth- Thirty-ninth 72nd 1/20th twentieth mid-19th thousandth 350th sixteenth 701st ...
PN	pronoun, nominal	none something everything one anyone nothing nobody everybody everyone anybody anything someone no-one nothin
PN\$	pronoun, nominal, genitive	one's someone's anybody's nobody's everybody's anyone's everyone's
PN+BEZ	pronoun, nominal + verb "to be", present tense, 3rd person singular	nothing's everything's somebody's nobody's someone's
PN+HVD	pronoun, nominal + verb "to have", past tense	nobody'd
PN+HVZ	pronoun, nominal + verb "to have", present tense, 3rd person singular	nobody's somebody's one's
PN+MD	pronoun, nominal + modal auxillary	someone'll somebody'll anybody'd
PP\$	determiner, possessive	our its his their my your her out thy mine thine
PP\$\$	pronoun, possessive	ours mine his hers theirs yours
PPL	pronoun, singular, reflexive	itself himself myself yourself herself oneseif ownself
PPLS	pronoun, plural, reflexive	themselves ourselves yourselves
PPO	pronoun, personal, accusative	them it him me us you 'em her thee we'uns
PPS	pronoun, personal, nominative, 3rd person singular	it he she thee
PPS+BEZ	pronoun, personal, nominative, 3rd person singular + verb "to be", present tense, 3rd person singular	it's he's she's
PPS+HVD	pronoun, personal, nominative, 3rd person singular + verb "to have", past tense	she'd he'd it'd
PPS+HVZ	pronoun, personal, nominative, 3rd person singular + verb "to have", present tense, 3rd person singular	it's he's she's
PPS+MD	pronoun, personal, nominative, 3rd person singular + modal auxillary	he'll she'll it'll he'd it'd she'd
PPSS	pronoun, personal, nominative, not 3rd person singular	they we I you ye thou you'uns
PPSS+BEM	pronoun, personal, nominative, not 3rd person singular + verb "to be", present tense, 1st person singular	I'm Ahm
PPSS+BER	pronoun, personal, nominative, not 3rd person singular + verb "to be", present tense, 2nd person singular or all persons plural	we're you're they're
PPSS+BEZ	pronoun, personal, nominative, not 3rd person singular + verb "to be", present tense, 3rd person singular	you's

PPSS+BEZ*	pronoun, personal, nominative, not 3rd person singular + verb "to be", present tense, 3rd person singular, negated	'tain't
PPSS+HV	pronoun, personal, nominative, not 3rd person singular + verb "to have", uninflected present tense	I've we've they've you've
PPSS+HVD	pronoun, personal, nominative, not 3rd person singular + verb "to have", past tense	I'd you'd we'd they'd
PPSS+MD	pronoun, personal, nominative, not 3rd person singular + modal auxiliary	you'll we'll I'll we'd I'd they'll they'd you'd
PPSS+VB	pronoun, personal, nominative, not 3rd person singular + verb "to verb", uninflected present tense	y'know
QL	qualifier, pre	well less very most so real as highly fundamentally even how much remarkably somewhat more completely too thus ill deeply little overly halfway almost impossibly far severely such ...
QLP	qualifier, post	indeed enough still 'nuff
RB	adverb	only often generally also nevertheless upon together back newly no likely meanwhile near then heavily there apparently yet outright fully aside consistently specifically formally ever just ...
RB\$	adverb, genitive	else's
RB+BEZ	adverb + verb "to be", present tense, 3rd person singular	here's there's
RB+CS	adverb + conjunction, coordinating	well's soon's
RBR	adverb, comparative	further earlier better later higher tougher more harder longer sooner less faster easier louder farther oftener nearer cheaper slower tighter lower worse heavier quicker ...
RBR+CS	adverb, comparative + conjunction, coordinating	more'n
RBT	adverb, superlative	most best highest uppermost nearest brightest hardest fastest deepest farthest loudest ...
RN	adverb, nominal	here afar then
RP	adverb, particle	up out off down over on in about through across after
RP+IN	adverb, particle + preposition	out'n outta
TO	infinitival to	to t'
TO+VB	infinitival to + verb, infinitive	t'jawn t'lah

UH	interjection	Hurrah bang whee hmpf ah goodbye oops oh-the-pain-of-it ha crunch say oh why see well hello lo alas tarantara rum-tum-tum gosh hell keerist Jesus Keeeerist boy c'mon 'mon goddamn bah hoo-pig damn ...
VB	verb, base: uninflected present, imperative or infinitive	investigate find act follow inure achieve reduce take remedy re-set distribute realize disable feel receive continue place protect eliminate elaborate work permit run enter force ...
VB+AT	verb, base: uninflected present or infinitive + article	wanna
VB+IN	verb, base: uninflected present, imperative or infinitive + preposition	lookit
VB+JJ	verb, base: uninflected present, imperative or infinitive + adjective	die-dead
VB+PPO	verb, uninflected present tense + pronoun, personal, accusative	let's lemme gimme
VB+RP	verb, imperative + adverbial particle	g'ahn c'mon
VB+TO	verb, base: uninflected present, imperative or infinitive + infinitival to	wanta wanna
VB+VB	verb, base: uninflected present, imperative or infinitive; hyphenated pair	say-speak
VBD	verb, past tense	said produced took recommended commented urged found added praised charged listed became announced brought attended wanted voted defeated received got stood shot scheduled feared promised made ...
VBG	verb, present participle or gerund	modernizing improving purchasing Purchasing lacking enabling pricing keeping getting picking entering voting warning making strengthening setting neighboring attending participating moving ...
VBG+TO	verb, present participle + infinitival to	gonna
VDN	verb, past participle	conducted charged won received studied revised operated accepted combined experienced recommended effected granted seen protected adopted retarded notarized selected composed gotten printed ...
VDN+TO	verb, past participle + infinitival to	gotta
VBZ	verb, present tense, 3rd person singular	deserves believes receives takes goes expires says opposes starts permits expects thinks faces votes teaches holds calls fears spends collects backs eliminates sets flies gives seeks reads ...
WDT	WH-determiner	which what whatever whichever whichever-the-hell
WDT+BER	WH-determiner + verb "to be", present tense, 2nd person singular or all persons pl.	what're

WDT+BER+PP	WH-determiner + verb "to be", present, 2nd person singular or all persons plural + pronoun, personal, nominative, not 3rd person singular	whaddya
WDT+BEZ	WH-determiner + verb "to be", present tense, 3rd person singular	what's
WDT+DO+PPS	WH-determiner + verb "to do", uninflected present tense + pronoun, personal, nominative, not 3rd person singular	whaddya
WDT+DOD	WH-determiner + verb "to do", past tense	what'd
WDT+HVZ	WH-determiner + verb "to have", present tense, 3rd person singular	what's
WP\$	WH-pronoun, genitive	whose whosever
WPO	WH-pronoun, accusative	whom that who
WPS	WH-pronoun, nominative	that who whoever whosoever what whatsoever
WPS+BEZ	WH-pronoun, nominative + verb "to be", present, 3rd person singular	that's who's
WPS+HVD	WH-pronoun, nominative + verb "to have", past tense	who'd
WPS+HVZ	WH-pronoun, nominative + verb "to have", present tense, 3rd person singular	who's that's
WPS+MD	WH-pronoun, nominative + modal auxiliary	who'll that'd who'd that'll
WQL	WH-qualifier	however how
WRB	WH-adverb	however when where why whereby wherever how whenever whereon wherein wherewith wheare wherefore whereof howsabout
WRB+BER	WH-adverb + verb "to be", present, 2nd person singular or all persons plural	where're
WRB+BEZ	WH-adverb + verb "to be", present, 3rd person singular	how's where's
WRB+DO	WH-adverb + verb "to do", present, not 3rd person singular	howda
WRB+DOD	WH-adverb + verb "to do", past tense	where'd how'd
WRB+DOD*	WH-adverb + verb "to do", past tense, negated	whyn't
WRB+DOZ	WH-adverb + verb "to do", present tense, 3rd person singular	how's
WRB+IN	WH-adverb + preposition	why'n
WRB+MD	WH-adverb + modal auxiliary	where'd

ANEXO G ABREVIACIONES Y GÉNEROS

Abreviación	Español	Inglés
abr	abreviatura	abbreviation
adj	adjetivo	adjective
adv	adverbio	adverb
Agr	agricultura	agriculture
Anat	anatomía	anatomy
antic	uso anticuado	old-fashioned use
Antrop	antropología	anthropology
aprox	aproximadamente	approximately
Arquit	arquitectura	architecture
art	artículo	article
Astrol	astrología	astrology
Astron	astronomía	astronomy
Astronáut	Astronáutica	Astronauts
Auto	automovilismo	motoring
aux	auxiliar	auxiliary
Av	aviación	aviation
Biol	biología	biology
Bot	botánica	botany
Cicl	ciclismo	cycling
comp	comparativo	comparative
cond	condicional	conditional
conj	conjunción	conjunction
Constr	construcción	construction
Cosm	cosmética	cosmetic
Cost	costura	sewing
Culin	cocina	culinary
def	definido	definite
defect	defectivo	defective form
dem	demostrativo	demonstrative
Dep	deportes	sport
det	determinante	determiner
dim	diminutivo	diminutive
Ecol	ecología	ecology
Econ	economía	economy
Educ	educación	education
Elec	electricidad	electricity
Ent	entomología	entomology
esp	especialmente	especially
Esp	España	Spain
Estad	estadística	statistic

etc	etcétera	etcetera
euf	eufemismo	euphemism
excl	exclamación	exclamation
f	sustantivo femenino	noun feminine
fam	uso familiar	familiar use
Farm	farmacia	pharmacy
Ferroc	ferrocarril	railways
fig	uso figurado	figurative
Fil	filosofía	philosophy
Fin	finanzas	finance
Fís	física	physics
Fot	fotografía	photography
fpl	sustantivo femenino plural	noun feminine plural
frml	uso formal	formal use
Ftb	fútbol	soccer
fut	futuro	future
GB	Gran Bretaña	Great Britain
gen	uso general	general use
Geog	geografía	geography
Geol	geología	geology
Geom	geometría	geometry
ger	gerundio	gerund
Gimn	gimnasia	gymnastics
Hist	historia	history
hum	uso humorístico	humorous use
imperf	imperfecto	imperfect
impers	impersonal	impersonal
Impr	imprensa	printing
Ind	industria	industry
indef	indefinido	indefinite
indet	indeterminado	indeterminate
indic	indicativo	indicative
Indum	indumentaria	clothing
infin	infinitivo	infinitive
Inform	informática	information technology
interr	interrogativo	interrogative
inv	invariable	invariable
irón	uso irónico	ironic use
irreg	irregular	irregular
Jur	derecho	legal
Lab	laboral	labour
Lam	Latinoamérica	Latin America
Ling	lingüística	linguistics
lit	uso literal	literal use

Lit	literatura	literature
Loc:	locución, locuciones	idiom
loc adj	locución adjetiva	adjective idiom
loc adv	locución adverbial	adverb idiom
m	sustantivo masculino	noun masculine
m, f/mf	sustantivo masculino y femenino	noun masculine and feminine
Mat	matemáticas	mathematics
Med	medicina	medicine
Meteor	meteorología	meteorology
Mil	militar	military
Min	minería	mining
Mit	mitología	mythology
mpl	sustantivo masculino plural	noun masculine plural
Mús	música	music
n	nombre	noun
Náut	náutica	nautical
neg	negativo	negative
neut	neutro	neutral
npl	sustantivo plural	noun plural
ofens	ofensivo	offensive
onomat	onomatopeya	onomatopoeia
Ópt	óptica	optics
Orn	ornitología	ornithology
Parl	parlamento	parliament
pers	personal	personal
pey	uso peyorativo	pejorative use
Petról	industria petrolera	oil industry
pl	plural	plural
Pol	política	politics
pos	posesivo	possessive
pp	participio pasado	past participle
pref	prefijo	prefix
prep	preposición	preposition
pres	presente	present
pron	pronombre	pronoun
ps	pasado simple	past simple
Psic	psicología	psychology
Quím	química	chemistry
Rad	radio	radio
rel	relativo	relative
Rel	religión	religion
sb	alguien	somebody
Seg	seguros	insurance

sing	singular	singular
Soc	sociología	sociology
sthg	algo	something
subj	subjuntivo	subjunctive
suf	sufijo	suffix
superl	superlativo	superlative
Taur	tauromaquia	bullfighting
tb	también	also
Teat	teatro	theatre
Téc	técnica	technical
Tel	telecomunicaciones	telecommunications
Ten	tenis	tennis
Tex	textil	textile
Tip	tipografía	typography
Trans	transporte	transport
TV	televisión	television
Univ	universidad	university
US	Estados Unidos	United States
usu	usualmente	usually
v	verbo	verb
v aux	verbo auxiliar	auxiliary verb
Vet	veterinaria	veterinary
vi	verbo intransitivo	intransitive verb
v impers	verbo impersonal	impersonal verb
vr	verbo reflexivo	reflexive verb
vtr	verbo transitivo	transitive verb
Zool	zoología	zoology
®	véase	see also
»	equivalente cultural	cultural equivalent
Ô	marca registrada	trade mark

ANEXO H CONTRACCIONES

Contracción	Expansión
i'd	i would i had i should
you'd	you would you had you should
he'd	he would he had he should
she'd	she would she had she should
it'd	it would it had it should
we'd	we would we had we should
they'd	they would they had they should
hasn't	has not
hadn't	had not
haven't	have not
weren't	were not
i'm	i am
you're	you are
we're	we are
they're	they are
won't	will not
wouldn't	would not
gonna	going to
wanna	want to
shoulda	should have should has
what's	what is
where's	where is
which's	which is
who's	who is
let's	let us
that's	that is
'cause	because
there's	there is