UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
PLANTEL ARAGON

" TRADUCCIÓN AUTOMÁTICA Y ARD, UN TRADUCTOR HOLANDÉS-ESPAÑOL "

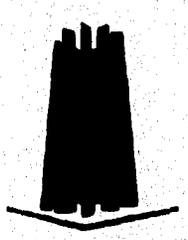
T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACIÓN

PRESENTA :

ARNIM RUELAS DÍAZ



DIRECTOR DE TESIS:

MTRA. ULRIKE TALLOWITZ PRADE

SEPTIEMBRE 1996

AGRADECIMIENTOS

A mis Padres:
por su apoyo, aliento y
todas las cosas que me han
enseñado en la vida.

A Joep, Paty y Saskia:
por su amistad, apoyo y
haberme enseñado el holandés.

A Ulrike y Randy:
por su paciencia y colaboración.

A July:
por su valiosa ayuda.

CONTENIDO

INTRODUCCIÓN

Existe dentro de la ciencia de la computación un área que pretende imitar el comportamiento humano a través de las computadoras, la inteligencia artificial. El procesamiento del lenguaje natural es uno de los principales campos de acción de esta área. En este campo se ubican los trabajos encaminados a simular las capacidades humanas que conciernen al lenguaje, y es aquí precisamente, donde la traducción automática tiene lugar.

Desde los primeros años de la aparición de las computadoras, el hombre pensó que podía imitar a través de éstas la acción de traducir un idioma (lenguaje natural) a otro. Los pequeños adelantos conseguidos al principio rápidamente se vieron obstaculizados tanto por la poca capacidad de las computadoras de esos días, como por la falta de una teoría lingüística adecuada. Con el paso del tiempo, el desarrollo tecnológico que ha hecho posible la creación de computadoras más veloces y con más capacidad de almacenamiento, además de los avances en la teoría lingüística han permitido desarrollar ciertos sistemas de traducción automática.

Básicamente, la traducción automática consiste en traducir textos de un idioma a otro, parcial o totalmente, a través de una computadora.

El presente trabajo pretende dar una visión general del campo de estudio de la traducción automática y desarrollar un pequeño sistema experimental de traducción holandés-español que ejemplifique algunos de los conceptos y problemas de la traducción automática. A este sistema se le ha denominado *Ard*. El nombre asignado al sistema no tiene un significado en especial, solamente se trata de un nombre propio de origen holandés.

El sistema está pensado en un idioma diferente del inglés debido a que en el mercado, actualmente sólo existen traductores que involucran el idioma inglés con algún otro. Al pretender hacer una traducción entre dos idiomas diferentes al inglés, por ejemplo, de holandés a español, primero se debe realizar una traducción del holandés al inglés y después otra del inglés al español. Hay que tener en cuenta que las traducciones hechas por los sistemas de traducción automática no son exactas, y que en cada proceso de traducción se

pierde información contenida en el original, por lo tanto al hacer dos traducciones para un mismo texto, en nuestro caso holandés-inglés e inglés-español, el resultado en español diferirá mucho del original en holandés. Es preferible desarrollar un sistema que directamente haga una traducción entre los dos idiomas, de manera que la información que se pueda perder en la traducción sea mucho menor. La selección del holandés como idioma para el traductor fue hecha en base a los conocimientos gramaticales que poseía del idioma.

El sistema *Ard* no pretende ser un traductor de todo el idioma holandés, es un sistema experimental que está basado en algunos cientos de palabras para ejemplificar la traducción de las diversas estructuras de oraciones incluidas en él, es decir, tiene más importancia traducir una mayor cantidad de estructuras de oraciones que una gran cantidad de palabras. También cabe mencionar que el sistema ARD traduce una sola oración a la vez. No tiene la capacidad de almacenar información contenida en oraciones anteriores para poder utilizarla en la traducción de alguna otra oración.

El sistema se desarrollará en una arquitectura IBM-PC bajo el lenguaje de programación Prolog. Este lenguaje de programación permite un manejo sencillo de las estructuras necesarias en Ard.

Para comprender mejor el funcionamiento de ARD, primeramente se expone en el capítulo I las bases en las que está fundada la traducción automática, así como su historia, los obstáculos que tiene que vencer y las principales estrategias de traducción usadas actualmente.

En el capítulo II, se muestra el porque de la selección de la estrategia de traducción, del modelo lingüístico, de los lenguajes de programación y de la arquitectura utilizada en ARD.

La implementación en el capitulo III, explica como se realizó la codificación de cada uno de los módulos utilizados en ARD, así como la forma de almacenamiento de la gramática y de las palabras de los diccionarios.

La forma de operar el sistema se encuentra detallada en el capitulo IV, en el cual se describen cada una de las pantallas y comandos utilizados en el sistema.

El capítulo V expone los alcances que se lograron con el sistema, lo que se puede desarrollar en un futuro, lo que no se pudo realizar y la forma en que se vencieron los obstáculos encontrados durante la elaboración del sistema.

Finalmente aparecen en el trabajo una serie de apéndices encaminados a auxiliar en la comprensión del sistema y a mostrar los resultados del programa. Entre ellos se encuentran un glosario con los principales términos lingüísticos y computacionales usados a lo largo de este trabajo; ejemplos de traducciones generadas por el sistema, de las reglas de gramática y de los diccionarios empleados en ARD.

I. TRADUCCIÓN AUTOMÁTICA

I. 1. Panorama General

Definición

El término *traducción automática* se refiere a la tarea de introducir un texto en un lenguaje natural (idioma fuente, SL -source language-) a un programa de computadora y producir un texto en otro lenguaje natural (idioma destino, TL -target language-), de manera que el significado del idioma fuente sea el mismo que el significado del idioma destino [Nirenburg 1992, p.8].

La traducción automática fue una de las primeras aplicaciones en que las computadoras fueron empleadas y ha sido una cuestión de debate desde entonces. La motivación para investigar más respecto a la traducción automática ha sido la necesidad de producir una mayor cantidad de traducciones, de forma más rápida y a un menor costo. Mucho del comercio mundial es conducido más allá de las fronteras nacionales y del idioma, así que cartas, contratos, manuales, acuerdos, etc., deben ser producidos en varios idiomas. En recientes años, esta necesidad de realizar más traducciones que las que los traductores humanos pueden producir, ha encaminado a un gran incremento en la actividad en este campo, especialmente en Europa donde la traducción de idiomas es un requerimiento para la Unión Europea.

El sueño y objetivo de muchos lingüistas y científicos ha sido desarrollar una Traducción Completamente Automatizada de Alta Calidad (Fully Automated High Quality Translation, FAHQT), la cual se daría si se pudieran hacer traducciones iguales a las que realizan los traductores humanos. Sin embargo, en la realidad no hay máquinas traductoras que con sólo oprimir unos cuantos botones, puedan tomar cualquier texto en cualquier idioma y producir una traducción perfecta a otro idioma sin intervención o ayuda humana. Este es un ideal para un futuro distante, lo cual muchos dudan.

Tipos de textos

Al traducir un texto, no hay sólo una traducción correcta para éste, sino que pueden darse una serie de traducciones correctas. Al realizar cada una de ellas se pueden tomar en cuenta diferentes especificaciones de traducción, como la decisión de retener la cultura original del texto o adaptarlo a la cultura del idioma destino, inventar traducciones para términos técnicos nuevos o dejarlos en el idioma fuente, mantener el mismo registro del documento fuente o cambiarlo por uno más sencillo, etc.

Las traducciones se realizan con textos de muy diversos tipos, por ejemplo, textos técnicos, legales o literarios. Los textos técnicos son generalmente informativos, las especificaciones de traducción tienen usualmente la misma terminología y una sintaxis simple. En cambio, la traducción de textos legales es muy difícil, debido a que se consideran textos muy específicos que difieren de acuerdo a los sistemas legales de cada país, además es necesario asignar equivalentes de traducción a unidades más grandes que sólo palabras y frases. En una traducción literaria, la intención e interpretación de ideas y sentimientos, además del estilo son lo importante, pero no existe un conjunto de especificaciones con las que se este de acuerdo para establecer con precisión estos puntos [Melby 1987, p. 145].

La traducción de tipo literaria como la de ciencia ficción o poesía no es una tarea para la traducción automática debido al carácter figurativo de sus textos y a la ambigüedad existente en la ideas que expresa. Existen otros tipos de textos adicionales que pueden ser considerados inapropiados para procesarlos automáticamente, por ejemplo, discursos de campañas políticas y anuncios publicitarios, debido al empleo de los juegos de palabras que los caracterizan, los cuales no pueden ser traducidos a otro idioma en la mayoría de los casos. Por lo tanto, el enfoque de la traducción automática se encuentra en la traducción de textos técnicos y otro tipo de textos informativos.

Traducción automática y otras ciencias

La traducción automática no es un campo de investigación pura, se puede decir que se encuentra entre los límites de la ciencia y la ingeniería. Toma de otras ciencias las teorías, ideas, métodos y técnicas que puedan servir para el desarrollo de los sistemas de traducción. Entre estas ciencias se encuentra la lingüística, de la que toma elementos de varias de sus ramas como la morfología, sintaxis, semántica, pragmática y análisis del discurso. De la lingüística computacional toma algoritmos para el análisis sintáctico de oraciones, para realizar interpretaciones semánticas y para la generación de textos. Otras ciencias que ayudan a la traducción automática son la ciencia de la computación; la teoría de la traducción; la inteligencia artificial, de la cual emplea los métodos de representación del conocimiento y la ingeniería de software, de esta última obtiene las técnicas de modularidad y construcción de bases de datos.

1.2. Elementos Lingüísticos

1.2.1. Niveles de descripción lingüística

La lingüística se encarga del estudio del lenguaje desde muy diversos aspectos, como su historia, evolución, adquisición, distribución, etc. Para la traducción automática, la parte importante es aquella relacionada con la estructura y el significado del lenguaje. Existen varios niveles lingüísticos mediante los cuales puede describirse el lenguaje, como se expondrá a continuación.

Ortografía

La ortografía se encarga de la forma correcta de escribir las palabras, lo cual es muy importante en la traducción automática, ya que ésta trata con textos. También cubre los problemas relacionados con la variación en el uso de los signos de puntuación de un idioma a otro. Estas variaciones deben ser tomadas en cuenta al implementar un sistema de traducción automática. Un ejemplo de variaciones ortográficas entre los idiomas son el uso de las comillas, en el francés se emplean los signos « y », mientras que en otros idiomas se emplean los signos ' y '. Otra diferencia es que el español usa dos signos de interrogación y admiración, mientras que otras lenguas no. En el alemán se usa una letra mayúscula como primer letra de un sustantivo; mientras que en idiomas como el español y el francés los

gentilicios van sin mayúscula. El uso de los puntos y las comas varía también de idioma a idioma, mientras que en el alemán es obligatoria una coma antes de una cláusula relativa, en el inglés es opcional.

Los sistemas de traducción que tratan con textos que poseen errores, pueden emplear los conocimientos de la ortotáctica –la cual especifica las secuencias de letras correctas en un idioma– para sugerir las correcciones de las palabras mal escritas. Por ejemplo, para el español existen secuencias validas como *br-*, *gr-*, *tr-*, *bl-*, *pl-*, pero secuencias como **dl-*, **nr-*, **ml-*, **sl-*, no son válidas. Para idiomas como el checo, la secuencia *zmrzl-* es válida, como en la palabra *zmrlzina* (helado) [Hutchins 1992, p.14].

Nota: El empleo del asterisco * es una convención para señalar oraciones o palabras que son agramaticales, inaceptables o anómalas.

Morfología

La morfología estudia la manera en que las palabras son formadas a partir de sus secuencias básicas o morfemas. Se divide en morfología inflexional y morfología derivacional.

La *morfología inflexional* define las posibles variaciones en las desinencias y raíces de las palabras de un idioma. Estas variaciones ocurren debido a las diferentes funciones gramaticales en que actúa una palabra, por ejemplo los sustantivos pueden ser singulares o plurales, los verbos tienen una conjugación diferente para cada persona y tiempo, etc.

El español es un idioma muy rico en cuanto a variaciones inflexionales, ya que los sustantivos varían en plural y singular, masculino y femenino; los verbos lo hacen en persona, tiempo y modo, mientras que los adjetivos lo hacen en plural y singular, masculino y femenino. Si se clasificaran los lenguajes de acuerdo al empleo de la morfología inflexional, en un extremo se encontrarían lenguajes como el chino que casi carece de este tipo de morfología y en el otro extremo se encuentran idiomas como el Eskimo en el cual el significado de una oración es expresado por las inflexiones en sustantivos y verbos. También existen lenguajes como el Turco en el cual se pueden añadir varios afijos inflexionales a una raíz [Hutchins 1992, p.15].

La *morfología derivacional* se encarga de la formación de palabras a partir de la unión de raíces y afijos a menudo realizando un cambio en la categoría gramatical. Por ejemplo a la raíz del sustantivo *amigo* se le agrega el sufijo *-able* y se obtiene al adjetivo *amigable*. También se emplea en la construcción de adverbios a partir de adjetivos. Agregando el sufijo *-mente* a los adjetivos, se construyen adverbios como *fácilmente, rápidamente*. En holandés a menudo se agrega el sufijo *-ing* a la raíz del verbo para formar un sustantivo, como el verbo *veroveren* (conquistar), cuya raíz *verover* forma el sustantivo *verovering* (conquista). Esta acción se conoce como nominalización.

Dentro de la morfología derivacional también se encuentra la composición de palabras, en la cual se unen palabras completas para formar palabras nuevas. Los significados de las palabras algunas veces son obvios debido a las partes que las componen, otras veces el significado es ligeramente diferente y en otras se forma una palabra totalmente diferente. En idiomas como el holandés y el alemán la formación de palabras de este tipo es muy común. Por ejemplo la palabra holandesa *brandalarm* (alarma de incendio), proviene de *brand* (incendio) y *alarm* (alarma). Pero las combinaciones pueden ser aún más complejas, empleando más de dos palabras como en *ontwikkelingssamenwerking* (cooperación para el desarrollo), formada por las palabras *ontwikkeling* (desarrollo), y *samenwerking* (cooperación), que a su vez está formada por las palabras *samen* (conjunta) y *werking* (actividad).

Sintaxis

La sintaxis es el estudio de la estructura de las oraciones, abarca las reglas o principios por los cuales las palabras se combinan para formar dichas oraciones. Estas reglas son aplicadas a las categorías gramaticales, las cuales se dividen en dos tipos, la primera consiste en elementos individuales del léxico, tales como sustantivo, verbo, adjetivo, preposición, etc.; la segunda esta formada por grupos de elementos tales como frase nominal, cláusula relativa, etc.

Las descripciones sintácticas son afectadas por tres tipos de relaciones en las oraciones: la *secuencia*, la *dependencia* y la *composición*. La *secuencia* se refiere al orden que siguen las categorías gramaticales dentro de la oración, por ejemplo, los adjetivos van después del sustantivo en español, mientras que en el inglés o el holandés ocurre lo contrario. La *dependencia* describe las relaciones entre categorías, es decir, la manera en que algunos componentes definen la forma de otros elementos en la oración. Un ejemplo de ello es el sujeto de una oración, el cual debe tener el mismo género y número que el verbo.

Por último, la *composición* se refiere a los elementos que forman parte de una categoría gramatical, por ejemplo, una frase nominal puede consistir de un artículo, un sustantivo y un adjetivo.

Las relaciones entre los componentes de una oración se establecen al compartir características sintácticas. La extensión de estas relaciones varía de idioma a idioma. En la oración *las perros blancos*, el artículo, el sustantivo y el adjetivo deben tener las mismas características sintácticas (género y número) para que la oración sea correcta, de lo contrario se produciría un error sintáctico, como en la oración **los perro blanco*. En el caso del inglés para esta misma oración, no hay posibilidad de cometer estos errores debido a que el artículo y el adjetivo no poseen formas distintas para el singular y el plural (*the white dogs*).

Las características sintácticas que llevan a cabo las relaciones entre los componentes se encuentran en el gobernador o cabeza, que es el elemento obligatorio en una estructura: el verbo en una oración, el sustantivo en una frase nominal, etc. Este gobernador o cabeza determina las formas de los elementos dependientes: los adjetivos deben concordar con el sustantivo, el sustantivo con el verbo, etc.

Las funciones gramaticales que desempeñan las palabras en las oraciones están relacionadas con los roles sintácticos que guardan con respecto al verbo, entre éstas se encuentran el sujeto, el objeto directo, el objeto indirecto, etc., que son parte importante del análisis sintáctico.

Semántica

La semántica es el estudio del significado tanto de las palabras en forma individual, ya sea aisladas o en el contexto de otras palabras, como de las oraciones. En esta fase de la traducción automática, se aplica sobre todo la semántica léxica o de palabras.

El significado de las palabras incluye características semánticas que representan atributos del mundo real. Algunas palabras comparten estas características, como por ejemplo las palabras *niño, hombre* y *mujer* comparten la característica "humano" y a su vez comparten con los animales la característica de "animado", lo cual los distingue de los objetos físicos "inanimados" (*hoja, casa, mesa*) o de conceptos "abstractos" (*belleza, inteligencia*). Las características semánticas a menudo son organizadas en jerarquías: por ejemplo, la característica semántica "humano" junto con la de "animal" forman la

característica "animado" y ésta junto con la de "vegetal" forman la de "ser vivo". Se pueden hacer generalizaciones en diferentes puntos de la jerarquía: por ejemplo, cualquier cosa animada puede ser sujeto del verbo caminar, pero sólo los "humanos" pueden escribir. Tales generalizaciones pueden ser hereditarias, de manera que basta con decir que la palabra *abuelo* tiene la característica "humano" para saber que también tiene las características "animado" y "ser vivo".

Es común estudiar las relaciones entre los elementos léxicos dentro de un campo o sistema semántico. Ejemplos de estos sistemas son el vocabulario de parentesco: *padre, madre, tío, primo, hijo*, etc.; los verbos de movimiento como *caminar, correr, viajar*, etc. En muchos casos el análisis de un sistema semántico puede ser hecho en términos de sus características sintácticas, debido a que las palabras dentro del mismo sistema a menudo tienen un comportamiento sintáctico similar o comparable. Por ejemplo, los sustantivos del grupo semántico "animado" usualmente son el sujeto del grupo semántico de los verbos de "movimiento".

Dentro de la semántica se encuentran también las diferencias de registro, que se refiere a las diversas formas de expresar un concepto en forma familiar, formal, etc. La palabra *automóvil*, por ejemplo, puede ser expresada en contextos más familiares como *carro* o *coche*.

Las diferencias en el significado de una misma palabra dependiendo del área de conocimiento en que se encuentre también son tratadas por la semántica. Por ejemplo el significado de la palabra *hoja* en botánica es diferente al empleado en la imprenta.

Análisis del discurso

El análisis del discurso estudia las relaciones que guardan las partes de un texto entre sí, es decir se analizan fenómenos más allá de la oración. Ejemplos de ello son las relaciones semántico-textuales y los conectores los cuales dan coherencia a un texto.

La *anáfora* consiste en las referencias hechas a un elemento mencionado explícitamente en algún lugar en el texto. Las referencias son hechas por pronombres, artículos, nominalizaciones, etc. En la oración *Pedro corre en el parque* se puede hacer una referencia a Pedro en una oración posterior dentro del texto por medio del pronombre *él*, por ejemplo, *él es muy veloz*.

Las referencias también pueden hacerse a elementos que no han sido mencionados con anterioridad en el texto, estos pueden ser elementos ya conocidos o que pueden ser fácilmente inferidos de la situación actual. También existen las referencias que se hacen a un elemento que se menciona más adelante en el texto. Estas referencias son muy raras y reciben el nombre de catáfora.

Los *conectores* sirven para ligar oraciones, frases, cláusulas o palabras dentro de un texto con el propósito de darle continuidad a las ideas. Una de sus funciones se encuentran la de compartir características de algunos elementos. En la oración *el libro rojo y viejo*, el conector *y* hace que las características *rojo* y *viejo* sean asociadas con el *libro*. Los conectores pueden unir también oraciones lógicamente, por ejemplo, el conector *porque* en la oración *no fuimos de día de campo porque estaba lloviendo* nos proporciona la causa por la que no se realizó la acción. También pueden cumplir una función restrictiva, por ejemplo, el conector *pero* en la oración *los niños irán de vacaciones pero sólo un día* cumple la función de limitar una acción.

Por último, la secuencia de palabras e ideas en las oraciones están determinadas por las relaciones que se encuentran en los textos. Por lo general, las primeras palabras de una oración hacen referencia a elementos ya mencionados o que se cree que ya son conocidos (*tema*). El resto de la oración contiene la información nueva acerca de esos elementos conocidos (*rema*).

1.2.2. Teorías de gramática y sus representaciones.

La gramática de un lenguaje es el conjunto de reglas que especifican las oraciones permitidas en ese lenguaje. Debe ser capaz de demostrar si una cadena de palabras cualquiera es parte o no de un lenguaje y debe asignar descripciones estructurales a las cadenas que son parte del lenguaje para explicar las relaciones entre las palabras. Estas gramáticas han sido descritas de diferentes maneras, creando así lo que se conoce como teorías gramaticales. Las teorías gramaticales siguientes son parte de los avances lingüísticos en las últimas tres décadas. Estas teorías son empleadas en los sistemas de traducción automática de forma individual o mezcladas. La ventaja de usar varias teorías en un mismo sistema es la de tomar ventaja de las capacidades de cada una de ellas para lograr una mayor precisión en las traducciones.

Gramática de estructura de frase

La gramática de estructura de frase (PSG, phrase structure grammar) fue desarrollada por Noam Chomsky en 1957, con el propósito de idear un enfoque matemático del lenguaje. Estas gramáticas consisten de una serie de reglas de reescritura de la forma:

$$A \rightarrow \alpha,$$

donde: A pertenece a un conjunto de símbolos no terminales

y α es una secuencia de símbolos terminales y no terminales.

Los símbolos no terminales representan estructuras sintácticas. Son parte de las reglas, pero no forman parte de las palabras del texto a analizar. Generalmente son escritos en letras mayúsculas. Por su parte, los símbolos terminales representan las palabras del texto a analizar y son escritos en letras minúsculas.

Las gramáticas de estructura de frase son parte de las gramáticas libres de contexto (CFG, context free grammar) debido a que A puede reescribirse como α independientemente del contexto en el que se encuentre. La aplicación de una secuencia de esta reglas de reescritura genera la representación de una oración. Por ejemplo la gramática de estructura de frase:

S → NP VP
VP → V NP
NP → PN
NP → ART N
V → escribió
PN → Juan
ART → la
N → carta

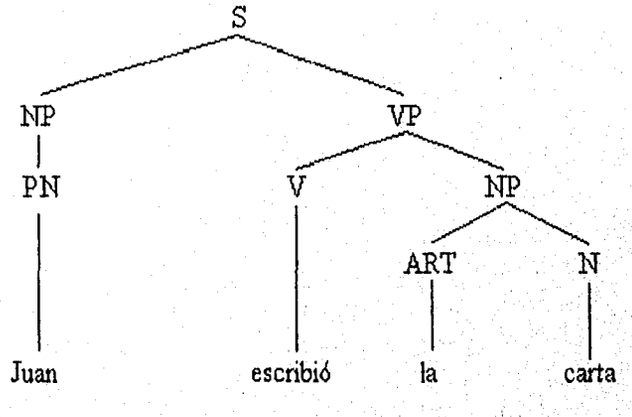
donde:

S	es una oración	Nota: Las abreviaturas utilizadas provienen del inglés por convención.
NP	frase nominal	
VP	frase verbal	
PN	nombre propio	
ART	artículo	
N	sustantivo o nombre	
V	verbo	

es capaz de generar la oración *Juan escribió la carta.* En esta gramática existe un símbolo de inicio representado por la letra S. Este símbolo de inicio es importante porque de ahí se comienza la generación de las oraciones permitidas por la gramática. Después se aplican las reglas hasta obtener la oración final.

S
NP VP
PN VP
Juan VP
Juan V NP
Juan escribió NP
Juan escribió ART N
Juan escribió la N
Juan escribió la carta

Este análisis puede ser representado por un árbol de estructura de frase:



donde puede verse también la manera en que se generó la oración. Tanto en la gramática como en el árbol se dan los principios de *dominancia* (en la primera regla S domina a NP y VP) y *precedencia* (NP precede a VP en la primera regla).

Lo que hace una PSG es definir una relación formal entre un texto y su representación abstracta. Esta gramática puede ser usada para darse cuenta de la representación que debería tener un texto o para producir textos aceptables. Las reglas de las PSG no son procedimientos para ser seguidos, sino que son definiciones y hasta cierto punto son reversibles.

Las PSG fueron y siguen siendo un principio muy importante aplicado a los analizadores gramaticales de los sistemas de traducción, pero se tuvo que buscar otro tipo de gramáticas debido a que no es posible representar algunas de las estructuras de los lenguajes naturales por medio de ellas.

Gramática de Cláusula Definida

La gramática de cláusula definida (DCG, Definite Clause Grammar) es un formalismo poderoso para la descripción de lenguajes formales y naturales. Creada por Colmerauer y Kowalski a partir de las gramáticas libres de contexto, la DCG tuvo el propósito de expresar las gramáticas libres de contexto en una forma lógica. Creando lo que se denominó cláusulas definidas o cláusulas de Horn.

Una regla de una gramática libre de contexto puede escribirse como una cláusula de la gramática de cláusula definida agregando un punto inicial y un punto final a cada uno de los elementos dentro de la regla. De esta forma tenemos que la regla de CFG:

$$S \rightarrow NP\ ,\ VP$$

puede ser escrita como una regla de la gramática de cláusula definida de la siguiente forma:

```
sentence(S0,S):-
        noun_phrase (S0,S1), verb_phrase(S1,S).
```

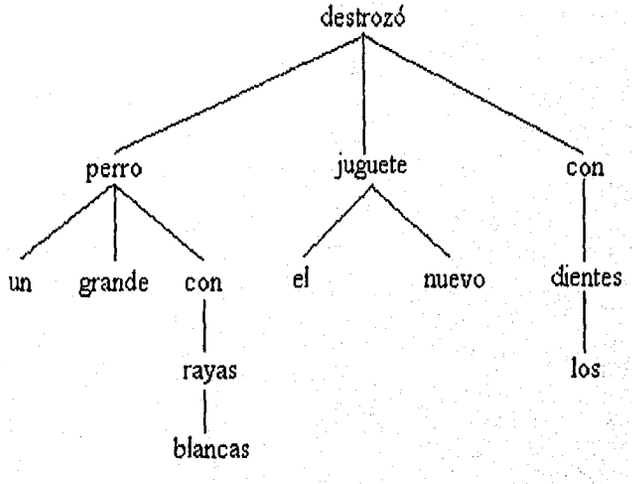
La cláusula puede leerse como: "La oración (sentence) se extiende de S0 a S siempre y cuando exista una frase nominal (noun_phrase) que se extienda de S0 a S1 y una frase verbal (verb_phrase) que se extienda de S1 a S".

La DCG también permiten la inclusión de condiciones extras a las reglas de la gramáticas. Esto permite generar una dependencia del contexto al establecer condiciones para que una palabra pueda seguir a otra dentro de una oración.

Relaciones de dependencia y gramática de valencia

Las relaciones de dependencia consisten en el establecimiento dentro de una oración de los gobernadores o cabezas y de los elementos dependientes. El verbo es el núcleo de una oración y por tanto el elemento gobernador de la misma. A partir de él se generan sus elementos dependientes como sustantivos y preposiciones que a su vez son gobernadores de otros elementos. Los sustantivos generalmente gobiernan a los elementos de una frase nominal: determinantes, adjetivos y preposiciones. Las preposiciones por su parte, gobiernan al sustantivo que es parte de la frase preposicional.

Un método tradicional de representar la relaciones de dependencia en una oración es hacerlo por medio de un árbol de dependencia, donde los elementos dependientes de cada gobernador son representados saliendo de él como ramas. El árbol de dependencia para la oración *un perro grande con rayas blancas destrozó el juguete nuevo con los dientes* es la siguiente:



Los adjetivos (grande, nuevo, blancas) y los determinantes (un, el y los), son gobernados por los sustantivos (perro, juguetes, rayas y dientes). Los sustantivos a su vez, son gobernados por preposiciones o verbos. El orden de las ramas puede ser establecido de acuerdo a reglas específicas, pero generalmente el gobernador se coloca en el centro, y los elementos dependientes en el orden en que aparecen en la oración.

En la teoría de dependencia se definen dos tipos de relaciones entre el verbo y otros elementos en la oración. La primera de ellas son los complementos o argumentos, cuya función y forma sintáctica es especificada por el verbo. La otra son los adjuntos, los cuales son opcionales y actúan como modificadores de la oración. En la oración *su mamá le dio una manzana a Carlos en la mañana*, los complementos están representados por las frases: *su mamá*, *una manzana* y *a Carlos* , mientras que el adjunto está formado por la frase *en la mañana*.

El número y naturaleza de los complementos que necesita un verbo están dados por su valencia. Los valores típicos de valencia fluctúan entre 0 y 3. Los verbos con valencia 0 son aquellos que tienen un sujeto ficticio, como el verbo *llover*. Los verbos monovalentes o de valencia 1 se refieren a los verbos intransitivos que sólo necesitan de un sujeto, como el verbo *dormir*. Los verbos bivalentes o de valencia 2 son aquellos que toman dos argumentos, pueden ser transitivos necesitando un sujeto y un objeto, como el verbo *escribir*, o necesitar un sujeto y una frase preposicional. Los verbos trivalentes o de valencia 3 incluyen verbos que necesitan un sujeto, un objeto directo y un objeto indirecto; un sujeto, un objeto y un complemento preposicional; o un sujeto y dos complementos preposicionales. Un ejemplo de este tipo de verbos es el verbo *dar* que necesita un sujeto, un objeto directo y un objeto indirecto. En la oración *él me dio el libro*, el sujeto corresponde al pronombre *él*, el objeto directo lo conforma la frase *el libro* y el pronombre *me* actúa como objeto indirecto.

Mientras que los complementos pueden ser controlados por el verbo, los adjuntos son menos predecibles y se pueden presentar con cualquier verbo. El verbo y sus complementos expresan la idea central de una oración, mientras que los adjuntos expresan circunstancias de tiempo, lugar, razón, propósito, etc., que acompañan a la oración central.

Además de describir la forma sintáctica de los complementos, una descripción de valencia también podría expresar las restricciones de características semánticas de los complementos, por ejemplo el verbo *caminar* necesita de un complemento (sujeto) animado, mientras que el verbo *leer* necesita de un complemento humano y uno leíble (que pueda ser leído).

La gramática de valencia es útil para la traducción en dos formas, primero porque los verbos generalmente tienen el mismo numero y tipo de valencia en diferentes lenguajes aunque tengan una estructura sintáctica diferente. De esta manera los argumentos pueden ayudar a resolver las ambigüedades que presentan los verbos polisémicos. El segundo

beneficio es la traducción de las preposiciones. Si la frase preposicional es un complemento, el verbo de la oración nos ayudará a traducir la preposición. Por el contrario, si la frase preposicional es un adjunto, la traducción de la preposición debe hacerse independientemente del verbo de la oración.

La valencia no sólo se aplica a los verbos y a sus argumentos. Los adjetivos también tienen patrones de valencia, especialmente cuando son usados de una manera predicativa, por ejemplo *interesado en*, *digno de*, etc.

Gramática de casos

Los principios de la gramática de valencia son empleados en la gramática de casos, creada en 1968 por C. Fillmore, en ella son establecidas las funciones semánticas (casos) de las frases nominales de una oración. La función semántica que desempeña una frase nominal(complementos y adjuntos) en su relación con el verbo gobernador de la oración se denomina el rol de la frase. Las funciones sintácticas pueden variar de lenguaje a lenguaje, mientras que las funciones semánticas permanecen constantes. Este tipo de gramática hace uso de los llamados "roles semánticos" o "roles temáticos".

El número de posibles funciones semánticas o roles varia de teoría a teoría, algunas emplean solamente 6 mientras que otras involucran más de 20. Winston establece los casos siguientes [Winston 1984, pp.314-316]:

Agente: Es el instigador de una acción. Es muy común que el agente corresponda con el sujeto sintáctico de la oración. *La niña* tiró el agua.

Objeto o Paciente: Es la entidad afectada por la acción. La niña tiró *el agua*.

Instrumento: Es la herramienta usada por el agente. María abrió la puerta *con la llave*.

Co-agente: Es un compañero del agente. Ambos desempeñan la acción. Arturo jugó cartas *con Alberto*.

Beneficiario: Es la persona para la que un hecho es desarrollado. Juan limpió el baño *para las visitas*.

Origen: Es la posición inicial de un cambio de estado, usualmente una posición física. El perro fue *de la sala* al patio.

Destino: Es la posición final del cambio de estado. El perro fue de la sala *al patio.*

Medio de Transporte: Es la entidad en la que se viaja. Mi tío vino *por avión.*

Trayectoria: Es el lugar por donde se desarrolla un movimiento. El avión voló *sobre la montaña.*

Localización: Es donde un hecho ocurre. Busqué el libro *en el escritorio.*

Tiempo: Especifica cuando ocurre el hecho. Juan realizó el examen *después de estudiar.*

Duración: Especifica cuanto tiempo dura el hecho. Juan estudió *durante dos horas.*

La designación de los roles puede ser auxiliada por las preposiciones, los verbos y los sustantivos. Cada preposición sólo admite ciertos roles, por ejemplo la preposición *con* sólo puede ser usada en los roles de co-agente o instrumento pero no los de trayectoria o beneficiario. Con cada verbo aparecen sólo determinados roles, el verbo *leer* puede emplear los roles de agente y objeto pero no el de trayectoria. Las características semánticas del sustantivo también nos auxilian en la determinación de los roles, por ejemplo, un instrumento debe ser algo inanimado.

El uso de la gramática de casos dentro de la traducción automática es útil en la resolución de ambigüedades en el momento de realizar las traducciones de verbos, sustantivos y preposiciones.

Principio de unificación y representación de características

Una vez expuestos los tipos de gramáticas a continuación se presentarán algunas representaciones formales.

Las características sintácticas y semánticas pueden ser representadas dentro de una gramática como atributos con sus valores correspondientes. Entre estas características se encuentran las categorías gramaticales (sustantivo, verbo, frase nominal), el género, el

número, el tiempo, las funciones gramaticales (sujeto, objeto), etc. Los nodos de las representaciones no sólo serán etiquetados con un valor, sino con características sintácticas y semánticas. Los conjuntos de tales características son llamados usualmente paquetes de características (feature bundles) [Hutchins 1992, p.25].

Este tipo de representaciones basadas en características, también incluyen una teoría de características, por medio de la cual se especifican las características empleadas, los posibles atributos de cada una y la forma en que estas características deben ser combinadas.

La oración *Juan escribió la carta* puede ser representada con sus paquetes de características como:

[cat: oración
{[cat: np, función: sujeto, número: singular, género: masculino,
{[cat: nom_prop, función: cabeza, número: singular, léxico: Juan,
semántica: {humano}]}]},
{[cat: vp, función: predicado,
{[cat: v, función: cabeza, tiempo: pasado, léxico: escribió]},
{[cat: np, función: objeto, número: singular, género: femenino,
{[cat: det, número: singular, género: femenino, léxico: la]},
{[cat: sust, número, singular, género: femenino, léxico: carta,
semántica: {inanimado}]}]}]}]

La unificación de características se logra cuando los atributos de éstas son compatibles, con lo que se forman unidades más grandes, hasta formar oraciones. Esto es conocido como principio de unificación. La compatibilidad puede tener dos casos, en el primero no hay características en común entre los elementos que se van a unificar, por lo que la unidad mayor se forma sin problemas:

mi [ ] $\cup$ gato [gen: masc, sem: animado] = mi gato [gen: masc, sem: animado]

En este caso la palabra *mi* no tiene un atributo de género, ni un atributo semántico por lo que la unidad mayor se forma con los atributos de la palabra *gato*.

Hay otro caso en el que los atributos de las características en común tienen los mismos valores, Por ejemplo en:

la [gen: fem] $\cup$ casa [gen: fem, sem: inanim] = la casa [gen:fem, sem: inanim]

La característica de género de ambas palabras tiene el valor femenino, lo que hace posible la unificación.

Si existen características en común dentro de los elementos a unificar con valores distintos, la unificación no puede llevarse a cabo:

el [gen: masc] $\cup$ silla [gen: fem, sem: inanim] = $\varnothing$

Dentro de la traducción automática el principio de unificación es utilizado en cada una de las etapas del proceso de traducción, por ejemplo los analizadores sintácticos lo utilizan al momento de formar las estructuras de las oraciones.

1.3. Elementos Computacionales

Existen algunos elementos dentro de la ciencia de la computación que son muy importantes en el desarrollo de sistemas de traducción automática y no se deben pasar por alto. Con el uso adecuado de estos, el desarrollo de los sistemas puede simplificarse mucho. En este apartado también se describirán algunos de los elementos proporcionados por la lingüística computacional para la traducción automática, como los analizadores sintácticos.

1.3.1. Datos, algoritmos y módulos

Todo sistema de traducción automática se compone del algoritmo o programa de traducción, el hardware requerido para su funcionamiento y los datos, que se dividen en reglas de gramática y diccionarios o lexicón. Es una buena idea mantener separados los datos de los algoritmos, de esta forma la parte del equipo de trabajo especialista en programación puede dedicarse al desarrollo de los algoritmos, mientras que los especialistas en lingüística pueden enfocarse en los datos. Una ventaja adicional es que es fácil reconocer el origen de los errores que se puedan presentar en las traducciones del sistema. Un error en una palabra mal escrita puede ser un error en los datos, mientras que un error en la estructura de una oración puede ser originado por los algoritmos o por los datos.

La tecnología en las computadoras se ha desarrollado principalmente en países de habla inglesa, debido a esta razón las computadoras no tienen problemas para tratar con las letras del alfabeto inglés. Pero como en la traducción están implicados dos o más idiomas, las computadoras tienen que tratar con letras de otros alfabetos. Al desarrollar un sistema es

muy importante proporcionar las facilidades para poder hacer uso de estas letras o crear una forma de representarlas. Ejemplos de esto son el uso de la letra "ñ" en español, la letra "ß" del alemán, o las letras nórdicas "ä , æ".

El dividir el proceso de traducción en varias etapas (ver I.4) y trabajar por separado en cada una de ellas facilita mucho el trabajo, además que hace más sencillo identificar la parte que puede estar trabajando mal. Cada una de estas etapas se les denomina módulos y generalmente se desarrollan y prueban por separado para después conjuntar todo el sistema. Otra ventaja de la utilización de los módulos es que el sistema puede mejorarse cambiando un sólo módulo sin afectar el desempeño del sistema en su conjunto. De esta manera es posible agregar nuevos idiomas o herramientas a un sistema de traducción automática. Es por ello que la aplicación de este principio de modularidad es vital para los sistemas de traducción automática.

1.3.2. Lexicón

La parte más grande en un sistema de traducción automática está compuesta por los recursos léxicos. Estos recursos se refieren a las palabras y modismos utilizados en los idiomas involucrados dentro del sistema. A estos recursos también se les denomina con el término *lexicón*.

Los componentes de un diccionario tradicional se encuentran por lo general en su forma base (por ejemplo, la forma base de los verbos es el infinitivo), aunque esto puede variar de idioma a idioma. El orden en el cual aparecen las palabras también cambia debido al orden de las letras en los alfabetos. En español la cuarta letra es la "ch", mientras que para el inglés es la "d". Los diccionarios tradicionales, además poseen otra clase de información como la pronunciación, categorías gramaticales, definiciones y alguna información etimológica y de estilo.

El lexicón en un sistema de traducción automática es ligeramente diferente. Algunos sistemas poseen sólo formas completas, es decir, listas de palabras tal y como ellas se encuentran en los textos. Un lexicón de este tipo en español, por ejemplo, posee dos elementos para un sustantivo, uno en singular (*gato*) y otro en plural (*gatos*). Esta opción no es muy adecuada para usarse con lenguajes que poseen una gran cantidad de variaciones inflexionales, por lo que es preferible introducir en el lexicón sólo las raíces y efectuar un análisis morfológico antes de realizar una consulta al lexicón.

El lexicón de los sistemas de traducción automática posee la información necesaria para los procesos sintácticos y semánticos: la categoría gramatical (verbo, adjetivo, sustantivo), el género, el número, la información semántica (sustantivo animado, verbo que necesita un sujeto humano),etc.

Un sistema de traducción automática debe incluir también los datos con las correspondencias entre los elementos léxicos en diferentes idiomas. Muchos sistemas separan la información sintáctica y semántica de un idioma en particular (lexicón monolingüe) de la información con las correspondencias léxicas entre los idiomas (lexicón bilingüe), esto se lleva a cabo principalmente en los sistemas de transferencia (ver 1.5.2).

Un punto que se debe considerar es el tamaño del lexicón. Los idiomas generalmente poseen varios miles de palabras y el tratar de incluirlas todas dentro de un sistema de traducción automática junto con su información sintáctica y semántica, se convierte en un volumen de información que puede traer problemas de almacenamiento. Pero aún y cuando se tenga la capacidad de albergar a varios miles de palabras, quizá el acceso a los datos sea muy lento. Lo conveniente es evaluar la amplitud del lexicón de manera que no sea pobre, pero que tampoco afecte el desempeño del sistema haciendo más lentas las consultas. Algunos sistemas dividen el lexicón en varios diccionarios, uno para los elementos más comunes, uno para el vocabulario raro o más especializado, uno para las formas irregulares, uno para las expresiones idiomáticas, etc. Otros sistemas dividen su lexicón en diccionarios separados por áreas de conocimiento, uno para medicina, uno para física, uno para economía, etc. El sistema TAUM-Météo (ver 1.8.3, pag. $$) es un claro ejemplo de ello, ya que sólo emplea diccionarios acerca de la predicción del clima.

1.3.3. Analizadores sintácticos

Dentro de la lingüística computacional, el algoritmo que es capaz de producir la estructura sintáctica de un texto utilizando una gramática y un lexicón como herramientas es conocido como *analizador sintáctico* o *parser*. La gramática determinará las combinaciones y secuencias de palabras aceptables. Generalmente las reglas de la gramática son expresadas como reglas de reescritura (ver 1.2.2.). Existen varios tipos de analizadores sintácticos, aunque generalmente se dividen en dos: analizadores Bottom-Up y analizadores Top-Down. En el campo de la traducción automática son empleados para establecer la estructura sintáctica de las oraciones que forman el texto que se va a traducir.

1.3.3.1. Analizadores bottom-up

En los analizadores Bottom-Up, el análisis es iniciado en las palabras. La estructura de la oración se va construyendo aplicando las reglas de reescritura que especifican las gramáticas de derecha a izquierda [Allen 1988, p.60]. Para el análisis de la oración *el niño tiró la comida*, las reglas de reescritura que definen la gramática se representan de la forma:

S ← NP VP (1)
NP ← ART N (2)
VP ← V NP (3)
ART ← el (4)
ART ← la (5)
V ← tiró (6)
N ← comida (7)
N ← niño (8)

El analizador inicia aplicando la regla (4) remplazando la palabra *el* con el símbolo no terminal ART, con lo que se obtiene:

ART niño tiró la comida

después la regla (8) cambia la palabra *niño* por N dando como resultado:

ART N tiró la comida

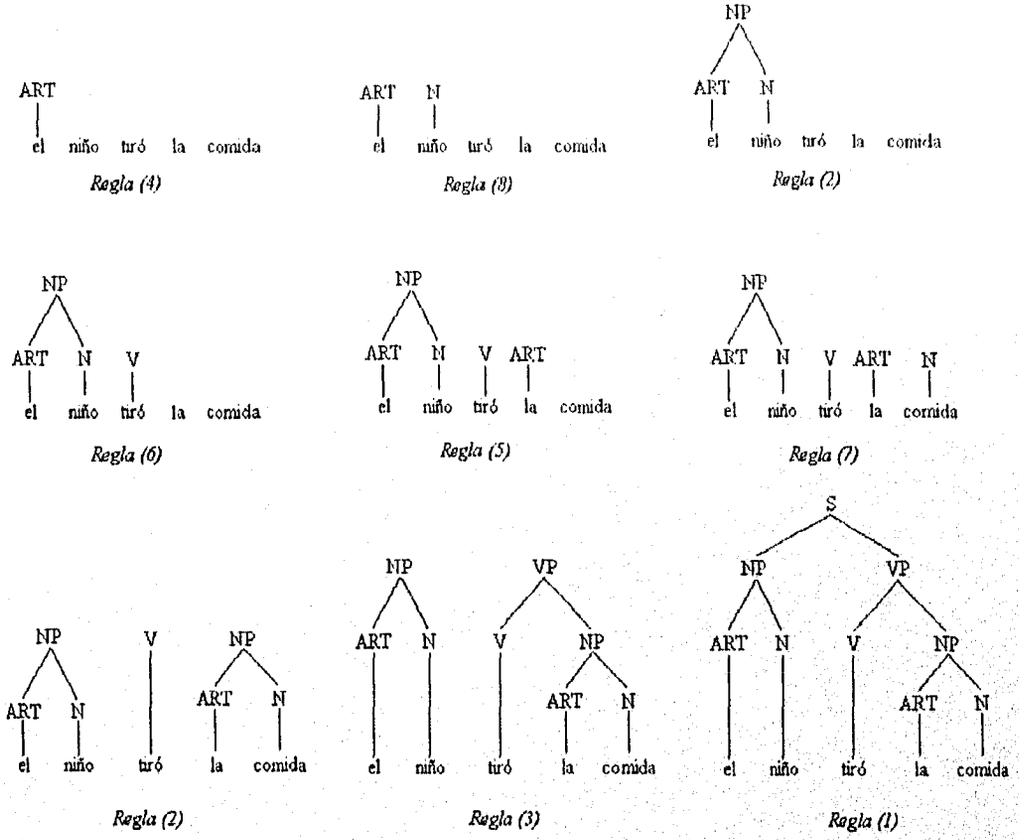
los dos primeros símbolos no terminales, según la regla (2), se pueden sustituir por NP, obteniendo:

NP tiró la comida

las reglas se continúan aplicando hasta que se obtiene la estructura completa de la oración, de esta manera, el análisis de la oración se completa con:

NP V la comida	regla (6)
NP V ART comida	regla (5)
NP V ART N	regla (7)
NP V NP	regla (2)
NP VP	regla (3)
S	regla (1)

La idea principal es encontrar una secuencia de símbolos que se encuentren del lado derecho de una regla. Estos símbolos representan una subestructura que es sustituida por la parte izquierda. Con el símbolo S, el análisis queda terminado, porque se ha llegado al símbolo que representa la oración, por lo tanto se establece que se trata de una oración válida para la gramática. Cada regla aplicada forma una estructura que es parte del análisis y puede ser representada mediante un árbol de estructura de frase:



El árbol resultante que muestra la estructura sintáctica de la oración es construido de manera ascendente, de ahí el nombre del analizador.

1.3.3.2. Analizadores top-down

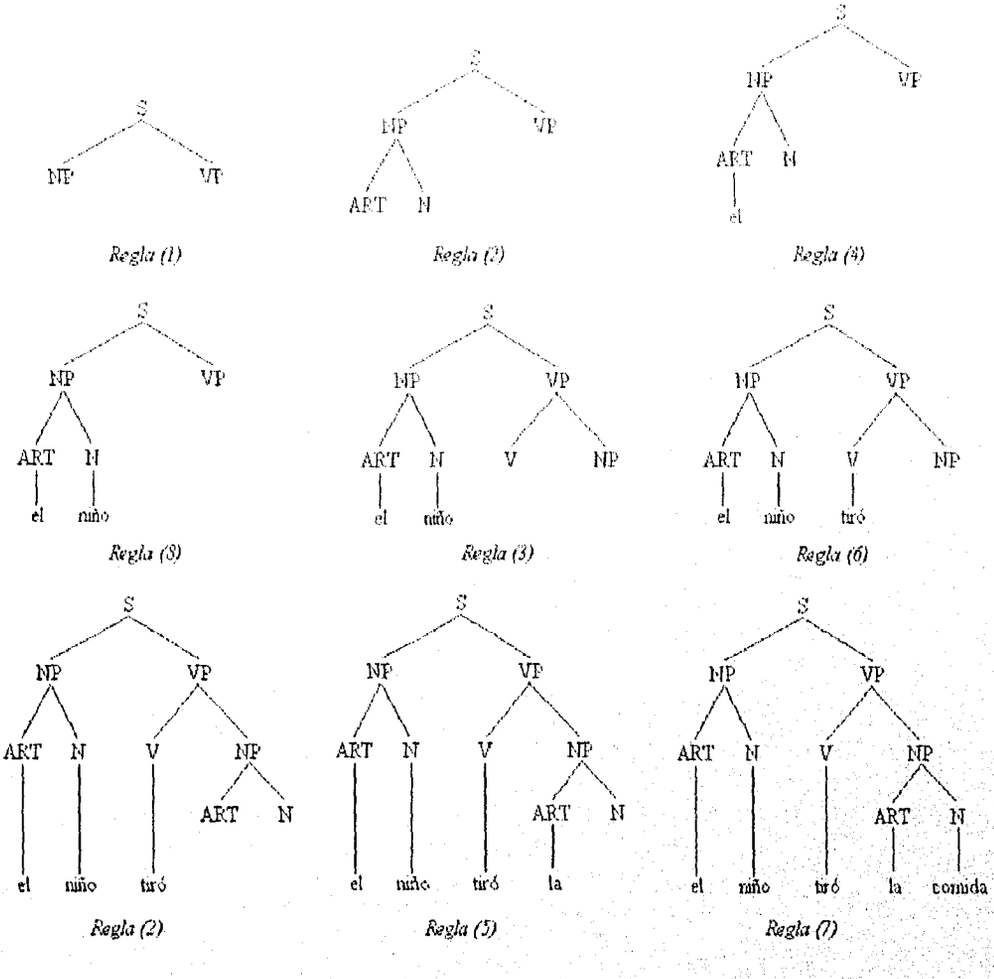
Los analizadores sintácticos Top-Down, al contrario de los Bottom-Up empiezan en un nivel abstracto con un símbolo inicial, a partir de este símbolo se forma una estructura sintáctica al aplicar las reglas de reescritura que forman la gramática. El análisis finaliza cuando se llega a las palabras que forman la oración. En este tipo de analizadores las reglas de reescritura son aplicadas de izquierda a derecha. Para un analizador Top-Down, la gramática usada en el ejemplo anterior se escribiría:

S → NP VP	(1)
NP → ART N	(2)
VP → V NP	(3)
ART → el	(4)
ART → la	(5)
V → tiró	(6)
N → comida	(7)
N → niño	(8)

La aplicación de las reglas para el análisis de la oración *el niño tiró la comida* es el siguiente:

S	símbolo inicial
NP VP	regla (1)
ART N VP	regla (2)
el N VP	regla (4)
el niño VP	regla (8)
el niño V NP	regla (3)
el niño tiró NP	regla (6)
el niño tiró ART N	regla (2)
el niño tiró la N	regla (5)
el niño tiró la comida	regla (7)

En este tipo de analizadores, el árbol que representa la estructura sintáctica de la oración se forma de arriba hacia abajo:

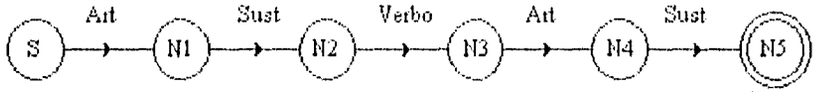


Actualmente, se emplean tanto los analizadores Bottom-Up como los Top-Down. La diferencia en la manera en que van construyendo la estructura sintáctica de la oración los hace apropiados para trabajar con tipos de estructuras diferentes.

1.3.3.3. Redes de transición

Existen otro tipo de analizadores sintácticos que están basados en las redes de transición, las cuales consisten en un conjunto de estados o nodos conectados por arcos. Cada arco está etiquetado con una categoría gramatical y representa una transición entre dos nodos. Empezando en un nodo inicial, un arco puede ser atravesado si la palabra que se está analizando pertenece a la categoría gramatical que se encuentra en el arco. Una vez atravesado el arco, se pasa a otro estado donde la palabra a analizar es cambiada por la siguiente palabra de la oración. Una oración es considerada como válida si es posible

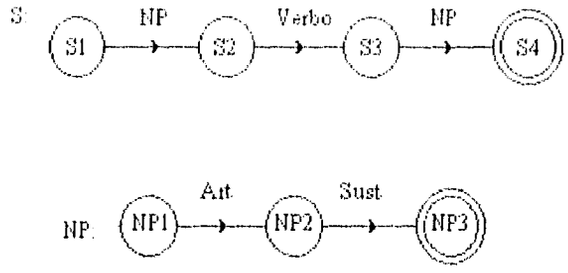
atravesar la red de transición desde el nodo inicial hasta el nodo final, sin que falten palabras por analizar en la oración [Allen 1987, p.44]. Los nodos generalmente son representados por círculos y los arcos por vectores dirigidos. El nodo final es representado de diversas formas, como un arco etiquetado con la palabra pop, como un nodo con un asterisco, etc. Para nuestro análisis utilizaremos un doble círculo. De esta forma, la red de transición que puede analizar la oración *el niño tiró la comida* es de la forma:



El análisis inicia en el nodo S, la primera palabra es un artículo por lo que se llega al nodo N1. La siguiente palabra es el sustantivo *niño*, lo hace atravesar el arco exitosamente hasta el nodo N2. El análisis continúa de la misma forma hasta que se llega al nodo N4, ahí la siguiente palabra a analizar es el sustantivo comida, lo que nos lleva al nodo final, como no quedan más palabras en la oración, se dice que se trata de una oración correcta.

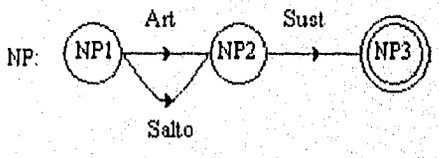
Redes de transición recursivas

Las redes de transición no son muy poderosas para describir un lenguaje, es por eso que se crearon las redes de transición recursivas (RTN, Recursive Transition Network). Tienen el mismo objetivo que las redes de transición, pero tienen dos características que aumentan su poder descriptivo. La primera de ellas es la capacidad de etiquetar los arcos no sólo con nombres de categorías gramaticales sino también con el nombre de otras redes, generalmente los arcos que designan otras redes se etiquetan con letras mayúsculas. Esta característica añade una capacidad de subrutinas a las redes de transición. Durante el recorrido de una RTN un arco que designa otra red puede ser atravesado solo si la red que representa puede ser atravesada. Debe tomarse en cuenta que una red puede hacer una llamada a si misma, haciendo lo que se conoce como una recursión verdadera. Para el ejemplo *el niño tiró la comida* se pueden utilizar las RTN de la siguiente manera:



El recorrido se inicia en la red S. El primer arco indica que se debe seguir la red NP. La red NP es atravesada debido a que las primeras palabras de la oración son un artículo y un sustantivo, al llegar al nodo NP3, el control se regresa al nodo S2, desde donde se continúa el recorrido de la red S. La siguiente palabra es el verbo *tiró*, con lo que se alcanza el nodo S3. Después se vuelve a hacer una llamada a la red NP. Las palabras *la* y *comida* hacen atravesar la red NP y el control vuelve al nodo S4 que es un nodo final. La oración es correcta debido a que se ha terminado el recorrido de la red principal y no han quedado palabras sin analizar.

La segunda característica de que añaden las RTN es la adición de arcos llamados arcos de "salto". Este tipo de arco puede ser recorrido sin necesidad de encontrar una categoría gramatical que lo defina y permite pasar de un nodo a otro incondicionalmente.



En esta red, para ir del nodo NP1 al nodo NP2, es opcional la existencia de un artículo, por lo que la red puede ser atravesada con sólo un sustantivo.

Aún con estas características añadidas, las RTN no son capaces de representar un lenguaje en su totalidad. Otro tipo de analizadores tuvieron que ser ideados para abarcar una mayor cantidad de estructuras.

Redes de transición aumentadas

Las redes de transición aumentadas (ATN, Augmented Transition Network), creadas por William Woods en 1970, tienen algunas características adicionales que las hace más poderosas que las RTN.

Las ATN emplean el uso de características sintácticas en el análisis de las oraciones. Establecen condiciones bajo las cuales un arco puede ser atravesado, es decir se debe cumplir con determinadas características para que un arco pueda ser atravesado. Añaden una serie de registros en los cuales se almacena información parcial que será utilizada en la formación de la estructura final del análisis. Vinculan ciertas acciones con los arcos, usualmente, la modificación de las estructuras.

1.3.3.4. Backtracking

En la práctica las gramáticas utilizadas no son tan sencillas como las que hasta ahora se han manejado, en algunas ocasiones hay que elegir entre varias alternativas posibles, por ejemplo en la gramática siguiente para un analizador top-down:

S → NP VP	(1)
NP → PRON	(2)
NP → PROP_N	(3)
NP → ART N	(4)
VP → V NP	(5)
PRON → yo	(6)
PROP_N → Juan	(7)
V → rompió	(8)
V → rompí	(9)
ART → el	(10)
N → plato	(11)

existen cuatro definiciones para una frase nominal (NP). Si se quiere analizar la oración *Juan rompió el plato*, el análisis inicia con la regla (1), la cual genera la secuencia NP VP. El siguiente paso es aplicar una de las reglas que designen una frase nominal. Para un humano es fácil saber que la regla que se debe de aplicar es la numero (3), debido a que la primera palabra de la oración es un nombre propio(PROP_N). Pero una computadora tiene que escogerla de alguna forma. Generalmente, cuando se presenta la opción de aplicar varias reglas -en este caso las reglas (2), (3) y (4)-, se aplica la que se presenta primero. El uso de

la regla (2) nos lleva a un camino en el que fallará el análisis, debido a que la regla (6) no concuerda con la primera palabra de la oración. Es aquí donde tiene lugar el concepto de *backtracking o vuelta atrás*, que consiste en guardar el estado que se tenía en el momento de usar una regla que tiene otras posibilidades, para en caso de que el análisis falle, se regrese al último punto de decisión y se tenga acceso a la siguiente opción. En nuestro ejemplo, utilizando el backtracking después de que la regla (6) falla, es posible regresar al punto donde se eligió la regla (2) y en esta ocasión elegir la regla (3). El análisis completo de la oración con el uso del backtracking se desarrolla de la siguiente manera.

S	Punto inicial
NP VP	Regla (1). Se establece el punto de decisión A.
PRON VP	Regla (2).
NP VP	Falla la regla (6). Se regresa al estado del punto de decisión A.
PROP_N VP	Regla (3).
Juan VP	Regla (7).
Juan V NP	Regla (5). Se establece el punto de decisión B.
Juan rompió NP	Regla (8). Se establece el punto de decisión C.
Juan rompió PRON	Regla (2).
Juan rompió NP	Falla regla (6). Se regresa al estado del punto de decisión C.
Juan rompió PROP_N	Regla (3).
Juan rompió NP	Falla regla (7). Se regresa al estado del punto de decisión C.
Juan rompió ART N	Regla (4).
Juan rompió el N	Regla (10).
Juan rompió el plato	Regla (11).

El concepto de backtracking es muy utilizado al implementar cualquiera de los analizadores sintácticos mencionados en un sistema de traducción automática. También cabe mencionar que el orden en que son puestas las reglas en el analizador es muy importante. Las reglas que se utilizan con más frecuencia deben ser colocadas al principio, para evitar gastar tiempo empleando el backtracking demasiadas ocasiones en el análisis de una oración.

la regla (2) nos lleva a un camino en el que fallará el análisis, debido a que la regla (6) no concuerda con la primera palabra de la oración. Es aquí donde tiene lugar el concepto de *backtracking o vuelta atrás*, que consiste en guardar el estado que se tenía en el momento de usar una regla que tiene otras posibilidades, para en caso de que el análisis falle, se regrese al último punto de decisión y se tenga acceso a la siguiente opción. En nuestro ejemplo, utilizando el backtracking después de que la regla (6) falla, es posible regresar al punto donde se eligió la regla (2) y en esta ocasión elegir la regla (3). El análisis completo de la oración con el uso del backtracking se desarrolla de la siguiente manera.

S	Punto inicial
NP VP	Regla (1). Se establece el punto de decisión A.
PRON VP	Regla (2).
NP VP	Falla la regla (6). Se regresa al estado del punto de decisión A.
PROP_N VP	Regla (3).
Juan VP	Regla (7).
Juan V NP	Regla (5). Se establece el punto de decisión B.
Juan rompió NP	Regla (8). Se establece el punto de decisión C.
Juan rompió PRON	Regla (2).
Juan rompió NP	Falla regla (6). Se regresa al estado del punto de decisión C.
Juan rompió PROP_N	Regla (3).
Juan rompió NP	Falla regla (7). Se regresa al estado del punto de decisión C.
Juan rompió ART N	Regla (4).
Juan rompió el N	Regla (10).
Juan rompió el plato	Regla (11).

El concepto de backtracking es muy utilizado al implementar cualquiera de los analizadores sintácticos mencionados en un sistema de traducción automática. También cabe mencionar que el orden en que son puestas las reglas en el analizador es muy importante. Las reglas que se utilizan con más frecuencia deben ser colocadas al principio, para evitar gastar tiempo empleando el backtracking demasiadas ocasiones en el análisis de una oración.

1.4. Etapas de la Traducción Automática

El proceso de la traducción automática comprende varias etapas. La aparición y tamaño de cada una depende del tipo de diseño que se emplee en el sistema (ver 1.5). En ciertos diseños algunas desaparecen, mientras que en otros no se pueden establecer con claridad sus límites.

Estas etapas son:

Entrada
Análisis
Transferencia
Generación o Síntesis
Salida

La etapa de *entrada* consiste en la lectura y almacenamiento del texto que se va a traducir, de manera que se encuentre disponible para la siguiente etapa, el análisis.

El *análisis*, en general, es la parte que reconoce la estructura y el significado de la oración que se va a traducir. Se caracteriza por emplear información solamente del idioma fuente sin importar el idioma destino. Se divide en varias etapas. La primera es el análisis morfológico, el cual abarca varias actividades: la primera es la identificación de palabras, que en la mayoría de los lenguas indoeuropeas es sencilla debido a que las palabras están separadas por espacios, no así en idiomas como el chino y el japonés. También es parte del análisis morfológico la identificación de las categorías gramaticales a las que pertenecen las palabras y la identificación de la forma en que se encuentran tales categorías gramaticales, por ejemplo la persona y el tiempo de un verbo, el género de un sustantivo, etc. Algunos sistemas de traducción utilizan este análisis para disminuir el tamaño de sus diccionarios. Al identificar la forma de las categorías gramaticales se pueden reconocer las raíces de las palabras y de esta forma hacer la consulta de diccionarios por medio de las raíces. La cantidad de sustantivos en el diccionario puede reducirse a la mitad almacenando un sólo elemento para el plural y el singular. Con los verbos, la reducción es mayor, porque al almacenar sólo la raíz, no será necesario almacenar cada una de las conjugaciones del verbo en cuestión. Otra actividad que se lleva a cabo dentro del análisis morfológico es la identificación de las palabras que forman parte de una palabra compuesta, principalmente en idiomas como el alemán y el holandés. Al identificar estos componentes, ya no es necesario el almacenamiento de las palabras compuestas. El análisis morfológico también puede ser

útil para reconocer las categorías gramaticales de palabras que no se encuentren en los diccionarios, a través de la identificación de las inflexiones gramaticales. Es conveniente mencionar que no todos los sistemas de traducción automática emplean el análisis morfológico, prefiriendo construir sus diccionarios con palabras en sus formas completas. Esto se debe principalmente a las irregularidades que se presentan en algunos idiomas en la formación de plurales, en la conjugación de verbos y aún más en la composición de palabras.

Posterior al análisis morfológico, se encuentra el análisis sintáctico el cual tiene como objetivo realizar las siguientes actividades: establecer las oraciones correctas en un lenguaje, determinar la estructura sintáctica al establecer la secuencia y el agrupamiento de los elementos gramaticales y reconocer las relaciones de dependencia entre los elementos de la oración. Se auxilia de los analizadores sintácticos y las gramáticas para realizar sus objetivos.

Algunos sistemas emplean una etapa de análisis semántico en el cual se intenta establecer elementos del significado de las palabras y frases para de esta manera resolver ambigüedades.

En la etapa de *transferencia* se llevan a cabo el cambio de los elementos léxicos y las estructuras sintácticas del idioma fuente a las del idioma destino. Estas transformaciones se realizan mediante reglas de transferencia léxicas (establecen los equivalentes de una palabra en otro idioma basados en ciertos datos) y estructurales (llevan a cabo los cambios en la estructura de la oración de un idioma a otro). Emplea información de los dos idiomas involucrados en la traducción.

La etapa de *generación o síntesis* obtiene las oraciones en el lenguaje destino a partir de las estructuras de representación elaboradas en la etapa de transferencia. Se caracteriza por emplear solamente información del lenguaje destino y se realiza independientemente del texto original.

La etapa de *salida* simplemente muestra y/o almacena el resultado de traducción.

1.5. Estrategias de Diseño

Al desarrollar un sistema de traducción automática es necesario elegir algunas de las estrategias de diseño disponibles. Estas estrategias abarcan varios aspectos y con la elección de cada una de ellos, el sistema a desarrollar presenta características peculiares.

1.5.1. Sistemas Bilingües o Multilingües

La primera decisión al diseñar un sistema es la de hacerlo bilingüe o multilingüe. Los sistemas que realizan traducciones entre un par de idiomas en particular son llamados sistemas bilingües. La traducción de este tipo es conceptualmente la más fácil y la necesidad de estas traducciones es posiblemente la más extendida.

Los sistemas bilingües pueden ser de dos tipos: unidireccionales o bidireccionales. Los primeros son aquellos que están diseñados para traducir de un lenguaje a otro sólo en una dirección. Mientras que los bidireccionales traducen en ambas direcciones.

Pero hay muchas situaciones en que la utilización de sistemas multilingües es necesaria. La traducción de varios idiomas a uno sólo es utilizada por organizaciones que reúnen y procesan información. La traducción de uno a varios idiomas es necesaria, por ejemplo, cuando un documento está destinado para ser usado en países con idiomas diferentes al original. Este caso es muy común en la Unión Europea, donde la traducción de documentos a varios idiomas es una actividad indispensable.

Los sistemas multilingües pueden traducir desde y hacia cada uno de los idiomas empleados o solamente realizar traducciones entre ciertos pares de idiomas. Un ejemplo del segundo tipo de sistemas es aquel en el que se involucran los idiomas español, inglés y holandés y que sólo puede hacer traducciones de los pares de idiomas español-holandés, español-inglés, holandés-español e inglés-español, sin que se lleve a cabo una traducción del holandés al inglés o del inglés al holandés. Este sistema podría ser empleado en una institución de habla española donde las traducciones entre el holandés y el inglés no son necesarias.

Los componentes de análisis y generación en un sistema multilingüe generalmente permanecen constantes sin importar los otros idiomas empleados en el sistema. Es decir, el proceso de análisis no debe cambiar dependiendo del idioma destino, ni el proceso de generación debe ser diferente al provenir de textos de idiomas distintos.

1.5.2. Estrategias de traducción

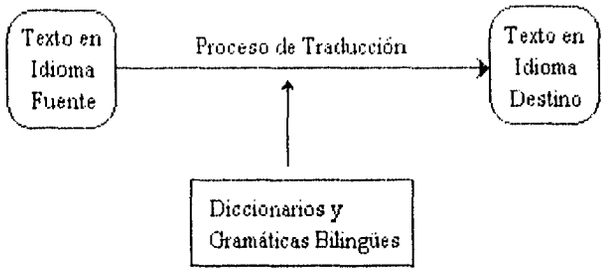
Las estrategias de traducción que han sido adoptadas para crear sistemas de traducción automática son la *estrategia directa*, la *estrategia de transferencia* y la *estrategia interlingua*. La elección de alguna de ellas determinará la estructura que tendrá el sistema a desarrollar.

Estrategia directa

La estrategia directa se le conoce como de primera generación porque fue la que históricamente surgió primero. Es empleada principalmente en sistemas bilingües unidireccionales. Se caracteriza por carecer de etapas intermedias en el proceso de traducción, el texto de entrada en el lenguaje fuente se pasa casi "directamente" al texto de salida en el lenguaje destino. Cada una de las etapas del proceso de traducción no están claramente definidas, sino que más bien se encuentran mezcladas de manera que no se pueden distinguir sus límites. Esta estrategia inicia con lo que se podría llamar una fase de análisis morfológico donde se lleva a cabo la identificación de las categorías gramaticales y algunas características de las palabras, como el número de los sustantivos, el tiempo de los verbos, etc. Esto se logra a través de la identificación de las terminaciones de las palabras. También se emplea la reducción de formas inflexionadas a sus formas básicas. No hay análisis de estructuras sintácticas ni de relaciones semánticas. Los resultados se llevan a lo que podría llamarse la etapa de transferencia, donde un programa busca en un diccionario bilingüe las equivalencias léxicas en el idioma destino. Posteriormente se aplican algunas reglas de reordenamiento que podrían ser parte de la etapa de transferencia y/o parte de la etapa de generación. Pueden ser parte de la transferencia debido a que las estructuras del lenguaje fuente se trasladan a estructuras del lenguaje destino y pueden ser parte de la generación porque sus resultados consisten de elementos del lenguaje destino en el orden del lenguaje destino.

Sus limitaciones son evidentes. Se trata de una traducción palabra por palabra con algunos reordenamientos. Las traducciones poseen errores a nivel léxico, las estructuras sintácticas son inapropiadas siendo muy similares a las del lenguaje fuente. La generación de las estructuras del lenguaje destino está basada en las estructuras del lenguaje fuente.

El proceso de la estrategia directa puede ser representada por medio del siguiente diagrama:



ESTRATEGIA DIRECTA

Desde un punto de vista lingüístico a esta estrategia le falta un análisis de la estructura interna del texto fuente, particularmente de las relaciones gramaticales entre las partes principales de las oraciones.

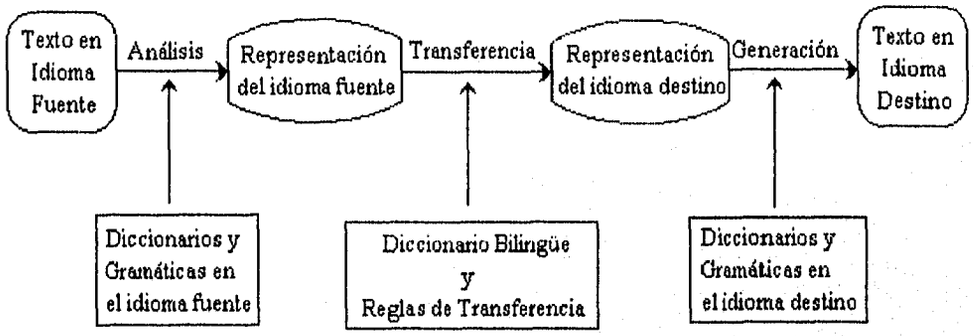
Estrategia de transferencia

La falla de los sistemas de primera generación inició el desarrollo de modelos lingüísticos de traducción más sofisticados. De esta investigación surgen los primeros sistemas indirectos o de la segunda generación conocidos también como sistemas interlingua y sistemas de transferencia.

Los sistemas de transferencia son aquellos en donde el resultado del análisis es una representación abstracta del lenguaje fuente y la entrada a la etapa de generación es una representación abstracta del lenguaje destino. En la etapa de transferencia de este tipo de estrategia se aprovechan las similitudes que existen entre los lenguajes involucrados en la traducción. La función de los módulos de transferencia bilingües es convertir las representaciones del lenguaje fuente en representaciones del lenguaje destino. A estas representaciones también se les llama representaciones interfaz porque unen módulos separados y son dependientes del lenguaje, es decir, que son específicas del lenguaje al que representan. Estas representaciones pueden variar de sistema a sistema, mientras que en algunos son sólo estructuras sintácticas, en otros pueden llegar a ser árboles sintáctico-semánticos, que conllevan un gran análisis para su creación.

Para llevar a cabo el proceso de traducción en esta estrategia son necesarios tres diccionarios: un diccionario del idioma fuente con la información sintáctica y semántica que es usado en el proceso de análisis, uno bilingüe utilizado en la etapa de transferencia y un tercero del idioma destino empleado en la etapa de generación.

El siguiente diagrama muestra el proceso de la estrategia de transferencia:



ESTRATEGIA DE TRANSFERENCIA

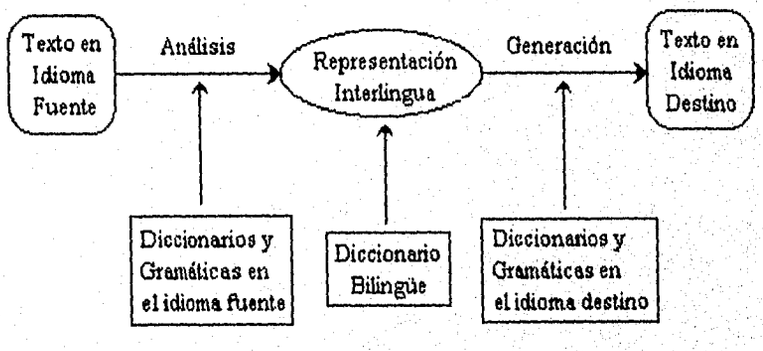
La transferencia se lleva a cabo en dos etapas: la transferencia léxica y la transferencia estructural. La transferencia léxica consiste en cambiar las palabras en el lenguaje origen por equivalentes en el lenguaje destino. La transferencia estructural se lleva a cabo cuando la estructura sintáctica del lenguaje fuente es inapropiada en el lenguaje destino. Esto se lleva a cabo mediante el uso de reglas de transferencia en las que se establecen las estructuras del lenguaje fuente y sus correspondientes en el lenguaje destino. Entre más profundo sea el análisis del lenguaje fuente, es menor la probabilidad de que aparezcan problemas, debido a que un análisis profundo tiende a neutralizar las distinciones entre los idiomas.

La desventaja de este tipo de sistema se presenta al agregar un idioma nuevo, porque al hacerlo es necesario implementar un módulo de análisis, uno de generación y los módulos de transferencia necesarios para poder traducir los idiomas ya existentes en el sistema. El número de módulos de transferencia necesarios para un sistema de n idiomas es $n(n-1)$.

Estrategia interlingua

Los sistemas interlingua se componen de dos fases: el análisis y la generación. La etapa de transferencia desaparece debido a que en el análisis el texto se lleva a una representación sintáctico-semántica intermedia común a varios lenguajes (interlingua), a partir de la cual el texto del lenguaje destino es generado. Esta representación debe incluir toda la información necesaria para la generación del lenguaje destino sin tomar en cuenta el lenguaje fuente, para lo cual se hace uso de diccionarios y gramáticas del idioma fuente. La generación incluye una fase sintáctica, donde se crean las estructuras del lenguaje destino, una fase semántica, en la que se resuelven las ambigüedades que no se resolvieron en el análisis, y una fase morfológica que realiza las inflexiones de las raíces. Para ello es necesario emplear los diccionarios y gramáticas del idioma destino.

El proceso de la estrategia interlingua, así como sus componentes pueden ser representados de la siguiente forma:



ESTRATEGIA INTERLINGUA

Este tipo de sistemas es muy conveniente para ser usados con sistemas multilingües. Cada módulo de análisis es independiente de los otros módulos de análisis y de los módulos de generación, por lo que es posible aumentar el número de idiomas con sólo agregar los módulos de análisis y generación correspondientes. Así, para tener un sistema que involucre n idiomas, capaz de hacer traducciones en $n(n-1)$ pares de idiomas, sólo son necesarios n módulos de análisis y n módulos de generación

Su principal desventaja es la dificultad de crear una representación interlingua capaz de incluir toda la información necesaria para realizar la generación de cualquier idioma que sea el destino.

Cada una de las estrategias se diferencia en el tamaño de los componentes de análisis, transferencia y generación. Existe un diagrama en forma de pirámide que muestra estas diferencias. El análisis del lenguaje se representa en el lado izquierdo de la pirámide y la generación en el lado derecho. La estrategia de traducción directa se presenta en la base de la pirámide con una etapa de análisis mínima, casi todo el trabajo es hecho en la etapa de transferencia. Hacia arriba se encuentra la estrategia de transferencia, en la cual, entre más profundo sea el análisis, más simple será su etapa de transferencia. En la cima de la pirámide se encuentra la representación interlingua que se alcanza a través de un análisis en el que sólo interviene el idioma fuente, hasta llegar al punto de no necesitar un etapa de transferencia para iniciar la generación.

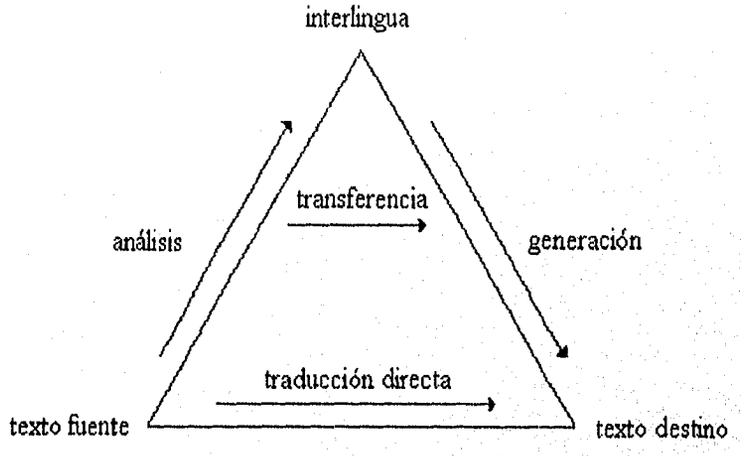


DIAGRAMA DE ESTRATEGIAS DE TRADUCCIÓN

1.5.3. La intervención humana

La tarea central de la traducción automática es la automatización de todo el proceso de traducción. Pero actualmente existen varios tipos de intervención humana dentro de este proceso que pueden ser añadidos a un sistema de traducción automática para mejorar la calidad de las traducciones hechas por el sistema. Puede ser empleado un sólo tipo de intervención humana o combinarlos para lograr mejores resultados. La desventaja de su uso es el tiempo que toma realizar cada una de ellas.

El primer tipo de intervención es la que ocurre antes del proceso de traducción y es conocida como *pre-edición*, la cual consiste en revisar el texto fuente para tratar de eliminar problemas con los que se pueda encontrar el sistema. Algunas de las acciones realizadas pueden ser: marcar los nombres propios, indicar las cláusulas dentro de las oraciones, marcar o sustituir palabras que no se encuentren dentro de los diccionarios del sistema, etc. También dentro de la pre-edición se encuentra el uso del lenguaje controlado en los textos fuentes, el cual reduce las ambigüedades potenciales y restringe la complejidad de las estructuras de las oraciones. El lenguaje controlado es capaz de tratar textos de cualquier área de conocimiento. Su objetivo es el de cambiar ciertas construcciones gramaticales que no pueden ser tratadas por el sistema.

La *post-edición*, se lleva a cabo después del proceso de traducción. Su tarea es la de revisar los resultados de la traducción automática para corregir errores y/o mejorar su calidad. La post-edición es hecha por traductores humanos de la misma forma que es hecha con las traducciones humanas. La persona debe conocer muy bien el tema tratado en la traducción y el idioma destino. Este tipo de intervención humana es muy utilizada actualmente en los sistemas comerciales para proporcionar traducciones que puedan ser comprendidas por cualquier persona. Hay ciertas circunstancias en las que la salida de los sistemas de traducción automática puede ser dejada sin editar, por ejemplo cuando está dirigida a especialistas familiarizados con el tema a tratar. Uno de los principales objetivos de las investigaciones en traducción automática es aumentar el poder de los sistemas de traducción de manera que la post-edición sea reducida y eventualmente eliminada.

Un tercer tipo de intervención humana ocurre en los *sistemas interactivos*. Los sistemas de este tipo se detienen durante el proceso de traducción para pedir ayuda a los usuarios con el propósito de resolver algunos problemas de interpretación de estructuras, resolución de ambigüedades y selección de equivalentes de traducción de palabras o frases. Un punto que se debe tomar en cuenta es la facilidad de manejo que debe presentar la interfaz para dar las opciones adecuadas de manera que no permita confusiones al operador humano.

Además de estos tipos de intervención humana existen algunas herramientas de traducción computarizadas que ayudan a los traductores humanos a realizar su trabajo. Estas herramientas pueden encontrarse en un sistema de traducción, aunque no sean incluidas dentro del término de traducción automática. Ejemplos de ellas son los diccionarios bilingües de consulta en línea, las bases de datos de terminologías, los lectores ópticos de caracteres, los revisores de gramática, ortografía y estilo, etc.

1.5.4. Sublenguaje

Otra ayuda que puede emplearse en el diseño de un sistema de traducción automática es el uso del sublenguaje. Consiste en el empleo de textos de una determinada área de conocimiento en un sistema de traducción automática. Al utilizar un sublenguaje se reducen la cantidad de elementos en el lexicón, porque cada área de conocimiento posee su propio lenguaje. Las ambigüedades se disminuyen debido a que la mayoría de las palabras solo tienen un posible sentido. El número de posibles construcciones sintácticas también disminuyen, porque un área de conocimiento generalmente posee construcciones sintácticas propias, las cuales sólo comprenden una parte de las construcciones sintácticas de un idioma. Al seleccionar el área de conocimiento se debe tener cuidado de que el área escogida no complique más la traducción, sino que sea un área que aumente las probabilidades de éxito del sistema. Por ejemplo, el involucrar el área legal a un sistema de traducción agrega un poco más de problemas a los ya presentes en el proceso de traducción.

1.6. Obstáculos en la Traducción Automática

La traducción automática se enfrenta con obstáculos que debe vencer, principalmente computacionales y lingüísticos. Los computacionales consisten en problemas con la capacidad de almacenamiento y con la velocidad de procesamiento de las computadoras en que son implementados los sistemas. Este tipo de problemas podrán ser resueltos en el futuro con los avances de la tecnología y la aparición de ordenadores más potentes.

Los problemas lingüísticos, por su parte, no son tan fáciles de resolver y son el principal punto de investigación de la traducción automática. Estos problemas pueden dividirse en problemas monolingües y problemas bilingües. Debido a su vital importancia en el proceso de traducción serán analizados en forma individual.

1.6.1. Problemas monolingües

Este tipo de problemas involucran un sólo idioma y se presentan principalmente en la etapa de análisis. Existe una diferencia entre los problemas morfológicos, léxicos y estructurales.

Problemas morfológicos y léxicos

En la etapa de análisis morfológico se dan una serie de problemas al tratar de reconocer la función de los afijos en las palabras. El primero de ellos son las irregularidades que se presentan en el uso de los afijos. Por ejemplo el español forma los plurales agregando los afijos *-s* o *-es* a la raíz de la palabra, pero existen palabras como *avestruz* en las que la raíz tiene que ser cambiada para formar el plural (*avestruc* + *-es*), e incluso existen palabras que terminan en *-s* y no representan ningún plural (*compás, arnés*). En otros idiomas, existen incluso sufijos que representan más de una función inflexional. como el sufijo *-en* del holandés que representa el plural de sustantivos, la terminación de los verbos en infinitivo y la terminación de las conjugaciones plurales de los verbos. Pero además existen sustantivos, en singular que terminan con *-en* y verbos en pasado participio que terminan en *-en*.

La determinación de los componentes de las palabras compuestas también originan ambigüedades. La palabra holandesa *filmtijdschift* puede ser dividida en *film* (cine) + *tijdschrift* (revista) o en *filmtijd* (hora de cine) + *scrift* (cuaderno).

Otro tipo de problema que se presenta en la etapa de análisis es conocido como homografía, en donde una palabra puede ser interpretada en más de una manera. Podemos diferenciar varias formas. En la primera, la ambigüedad se presenta en la categoría gramatical, a una palabra le pueden ser asignadas más de una categoría gramatical. Para un sistema de traducción automática es necesario saber que categoría gramatical posee cada palabra para poder hacer una traducción adecuada de ella. En español se presentan este tipo de ambigüedades en algunos adjetivos que pueden ser también usados como sustantivos, por ejemplo los colores(*verde, azul, blanco*) y los gentilicios (*argentino, belga, mexicano*). La palabra *sobre* puede ser utilizada como preposición o como sustantivo, mientras que las palabra *cerca* como sustantivo o como adverbio. En holandés hay palabras que pueden ser verbos y sustantivos, como por ejemplo *schaatsen* (patinar y patinaje), *denken* (pensar y pensamiento). El idioma con más ejemplos de este tipo es el inglés, en donde una palabra puede ser empleada como varias categorías gramaticales: los verbos generalmente son sustantivos, como *change* (cambiar y cambio)o *drink* (beber y bebida), los sustantivos pueden ser usados como adjetivos que modifican a otros sustantivos, por ejemplo la palabra *stone* (piedra) puede ser empleada como sustantivo: *the big stone* (la piedra grande) o como un adjetivo: *the stone wall* (el muro de piedra). Pero hay palabras que pueden ser incluidas en más de dos categorías gramaticales: *light* puede ser un adjetivo (ligero), un verbo (encender) y un sustantivo (luz), *last* puede ser un verbo(durar), un sustantivo(último), un adjetivo (último) y un adverbio(finalmente). Quizá el ejemplo más extremo es el uso de

palabras como *round* que pueden ser un sustantivo(ronda), un verbo(redondear), un adjetivo(redondo), una preposición (alrededor de) o un adverbio(circularmente). En todos los casos, el contexto juega un papel importante en su desambiguación.

Las ambigüedades léxicas también se presentan en lo que los lingüistas han denominado homonimia y polisemia. La *homonimia* se refiere a palabras que tienen la misma forma pero tienen significados muy diferentes, ejemplos de ello son las palabras *banco* (institución financiera y mueble) y *vela*(parte de un barco y utensilio de cera). La *polisemia* son palabras que tienen significados relacionados unos con otros, como *pata* (de un animal y de un mueble). Para efectos prácticos de la traducción automática, la diferencia entre homonimia y polisemia no es importante. El problema consiste en conocer el significado de las palabras según el contexto, para poder dar una traducción adecuada. El mismo contexto puede ayudarnos a resolver estas ambigüedades, por ejemplo, las oraciones que contienen la palabra *banco*, en el sentido de institución financiera, utilizan verbos como *abrir* y *cerrar*. Este tipo de verbos no es muy común que sean puestos en la misma oración junto con la palabra *banco* en el sentido de mueble.

Problemas estructurales

Este tipo de problemas tratan con las estructuras sintácticas que representan las oraciones. Se dice que son ambiguas cuando puede haber más de una estructura resultante de la etapa de análisis. Un claro ejemplo de este tipo de ambigüedades es el análisis de frases preposicionales. Cada frase preposicional puede modificar al verbo o a una frase nominal que le precede. El analizador sintáctico debe ser capaz de establecer estas relaciones para poder generar posteriormente una traducción adecuada. Por ejemplo en las oraciones:

El estudiante observó el cielo con un telescopio
El estudiante observó el paisaje con árboles

las frases preposicionales están modificando a componentes diferentes. En el primer caso la preposición *con* modifica al verbo *observó*, mientras que en la segunda modifica a la frase nominal *el paisaje*.

Otro ejemplo de ambigüedades estructurales es la *coordinación*. Consiste en establecer el alcance de los conectores de coordinación dentro de una oración. Un ejemplo muy común es el uso del conector *y*. En la oración *Juan limpió sus libros y cuadernos viejos*, el adjetivo *viejos* puede modificar sólo al sustantivo *cuadernos* o a *libros y*

cuadernos. Para precisar el alcance de estas relaciones el análisis tiene que ser más profundo, lo que evita encontrar problemas al realizar las traducciones a otros idiomas.

1.6.2. Problemas Bilingües

Los problemas bilingües tienen que ver con las equivalencias tanto del léxico como de las estructuras en los idiomas involucrados en la traducción.

Problemas léxicos

Uno de estos problemas se presenta cuando un concepto en un idioma corresponde a varios conceptos y por tanto varias palabras en otro lenguaje. Esto se presenta no porque la palabra en el idioma fuente sea ambigua, sino porque es ambigua desde la perspectiva del otro lenguaje. Ejemplos de ello son: la palabra española *rentar* que tiene dos equivalentes en holandés *huren*(pagar el dinero de la renta) y *verhuren*(prestar el servicio de renta). La palabra inglesa *leg* tiene los equivalentes en español *pierna* o *pata*. Algunas de estas ambigüedades se presentan debido a diferencias culturales.

Otro de los problemas que se presentan es la falta de correspondencias para algunas palabras, es decir, existen conceptos que no tienen una palabra correspondiente en el idioma destino, por ejemplo el verbo holandés *kluun* no tiene equivalente en español, pero significa algo como "caminar con patines donde no se pude patinar".

La selección de una palabra dependiendo del registro del documento en el lenguaje destino, es otra ambigüedad que se tiene que resolver. Por ejemplo la palabra inglesa *car* puede traducirse como *coche* en un registro familiar o como *automóvil* en un registro más formal.

Problemas estructurales

Por su parte, los problemas estructurales se presentan debido a las diferencias que existen entre las estructuras sintácticas de los idiomas involucrados en la traducción. Por ejemplo los adjetivos preceden al sustantivo en el inglés, pero van después del sustantivo en español. La estructura de la oración que muestra como se llama una persona es diferente en el inglés, el español y el holandés:

His name is John (inglés)
Se llama Juan (español)
Hij heet Jan (holandés)

En inglés se utiliza un posesivo, el verbo *is* (es) y el nombre. El español emplea un verbo reflexivo y posteriormente el nombre, mientras que el holandés emplea un verbo transitivo. Estas diferencias deben ser reflejadas por medio de reglas en el módulo de transferencia o en otro tipo de gramáticas que permitan hacer los ajustes necesarios a las estructuras.

Otro problema importante que debe ser considerado dentro de los sistemas de traducción automática es la traducción de los modismos y expresiones idiomáticas. Su importancia radica en el hecho de que sus traducciones no pude ser derivadas de las palabras que los componen, sino que deben ser traducidos como una sola unidad.

1.7. Criterios de Evaluación

Actualmente, los sistemas de traducción automática no alcanzan un desempeño ideal que consiste en introducir cualquier texto en cualquier idioma y mediante un proceso automático producir traducciones de alta calidad. Debido a esto, es necesario una metodología para evaluar el desempeño y eficiencia de los sistemas. Pero hasta hoy, a pesar de los más de 40 años de investigación en traducción automática, no se tiene una metodología definida. En muchas ocasiones las evaluaciones se llevan a cabo con metodologías que no se dan a conocer, mediante criterios muy subjetivos como la fidelidad o la precisión, o son hechas por personal que tiene muy poca o ninguna experiencia con las técnicas de traducción automática, por lo que sus evaluaciones no son realistas. También es necesario tomar en cuenta que una "buena traducción", ya sea producida por un humano o por una máquina, es un concepto difícil de definir. Mucho depende de a quien va dirigida y de las circunstancias particulares en las cuales es hecha, lo que agrega un punto más a lo subjetivo que puede ser una evaluación.

Sergei Nirenburg [Nirenburg 1992, pp.16-19] presenta algunos criterios comprensibles para evaluar el alcance, calidad de traducción, utilidad y extensibilidad de un sistema de traducción automática, los cuales son:

Alcance lingüístico. Se refiere al número de idiomas fuente y destino cubiertos por el sistema y la cobertura en la gramática y el vocabulario general. La cobertura gramatical puede ser calculada como un porcentaje de los fenómenos gramaticales que son analizados por el sistema.

Alcance de las áreas de conocimiento. Es el número de áreas de conocimiento cubiertos por el sistema y la amplitud de la cobertura de cada una. Entre más grande sea la cantidad de áreas de conocimiento que un sistema pueda cubrir, mayor será la utilidad general del sistema.

Grado de automatización. Entre menor sea el grado de intervención de los humanos en el proceso de traducción, mejor será el sistema, si la calidad de la traducción no cambia. Si añadimos la cantidad de tiempo requerida para la intervención humana y la dividimos entre el total de tiempo requerido para traducir el mismo texto manualmente, entonces se podrá establecer el grado de automatización como la proporción de mejora en la eficiencia de un sistema de traducción automática.

Precisión semántica. Es el grado en que un texto traducido expresa el mismo significado que el texto original. Las mediciones deben hacerse en los cambios sutiles más que en los grandes errores.

Inteligibilidad. Es el grado en que un texto traducido puede ser comprendido por lectores del idioma destino sin tener acceso al texto fuente. Esta medida debe hacerse en forma similar al punto anterior. Cuando estos dos criterios son combinados suele llamársele precisión general o calidad de la traducción.

Conveniencia. Se refiere al grado en que un texto es estilísticamente apropiado para el público al que está destinado. Por ejemplo la selección de elementos del léxico en un registro formal, pueden ser semánticamente exactos, pero inapropiados si el público final no comprende este vocabulario.

Portabilidad del idioma. Es la facilidad con que un nuevo idioma puede ser agregado. Ésta puede ser medida en tiempo de desarrollo humano empleado para añadir las reglas de gramática o elementos del léxico. En los sistemas con diccionarios bilingües, cada término del idioma fuente debe ser proyectado con su término correspondiente en todos los idiomas destino.

Mejoramiento. Es el grado en el que un sistema permite cambios y mejoras para hacerlo más automático y más preciso lingüísticamente sin comprometer considerablemente la calidad de la traducción.

Ergonomía. Se refiere a la facilidad de comunicación entre el usuario y el sistema de traducción automática. Esta comunicación debe ser clara y con una mínima oportunidad de cometer errores. En un proceso de traducción automática con intervención humana, convenientes medios de interacción entre los humanos y la máquina deben ser provistos para facilitar y hacer más rápido el proceso de traducción en general. Estas interfaces también deben incluir varias ayudas para los usuarios humanos, como diccionarios y enciclopedias.

Integrabilidad. Se refiere al grado en que un sistema de traducción automática puede ser un componente integral de un sistema de procesamiento de información como sistemas de recuperación de datos, lectores ópticos de caracteres, sistemas de procesamiento de mensajes, producción de textos, publicidad, etc.

Portabilidad de Software. Se refiere a la facilidad con que los sistemas de traducción automática pueden ser llevados a otras plataformas de hardware u otros sistemas operativos.

Para Hutchins y Somers [1992, p.162], los sistemas deben ser evaluados en las diversas etapas de su desarrollo. Las primeras evaluaciones son llamadas evaluaciones de prototipo. En ellas se evalúa el diseño básico del sistema, la escritura de programas, la compilación de diccionarios, etc. Estas evaluaciones deben ser hechas por el desarrollador del sistema, y debe evaluar cada uno de los procesos por separado. El objetivo de estas es conocer si los programas escritos para el sistema, realmente se están desempeñando en la forma que se espera y descubrir si la salida del sistema puede ser una "traducción aceptable".

En la siguiente etapa se llevan a cabo las evaluaciones de desarrollo, las cuales evalúan no sólo las capacidades lingüísticas, sino además otras capacidades. Los puntos a evaluar son el diseño y la provisión de facilidades para la entrada de textos, para revisar los resultados y para interactuar con la computadora. También incluye el tamaño y la eficiencia computacional de los programas y la integración del sistema a un medio ambiente en particular, es decir, la adaptación a una sistema operativo o a un hardware en particular. En esta etapa los desarrolladores deben asegurarse de cuáles son las capacidades reales del sistema.

La tercera etapa son las evaluaciones operacionales, que son llevadas a cabo por los compradores. En ellas se evalúan el desempeño y el costo-eficiencia del sistema. Las condiciones de trabajo, la clase de operadores que son requeridos, y qué tipo de facilidades técnicas ofrece el sistema.

Además Hutchins y Somers proponen una evaluación lingüística que se aplica a cada una de las fases citadas anteriormente y se basa en evaluaciones de calidad e identificación de errores. Los puntos considerados para evaluar la calidad son la fidelidad, inteligibilidad y estilo. La identificación de errores, por su parte, consiste en obtener el porcentaje de errores que aparecen en un texto, aunque esto no es muy preciso debido a que no se puede precisar lo que se considera como un error. Es preferible una clasificación de los errores y la cantidad de los mismos, lo cual da una mejor visión del desempeño del sistema.

1.8. Historia y Sistemas Desarrollados

Una forma de comprender el estado actual de la traducción automática se logra a través de ver su desarrollo en la historia y analizar los sistemas creados a lo largo de la misma. Convencionalmente, se distinguen tres etapas muy importantes: la etapa pre-ALPAC, el reporte ALPAC, la etapa post-ALPAC. Pero aún antes de la invención de las computadoras existieron algunos mecanismos para auxiliar en la traducción de idiomas. En 1933 se patentaron los primeros en Francia y Rusia. El primero de ellos fue patentado por Georges Artsrouni. Se trataba de un dispositivo que aceptaba por medio de un teclado una palabra en un lenguaje origen y proporcionaba sus equivalentes en varios lenguajes. Mediante un motor eléctrico se almacenaba y leía la información escrita en una banda de papel ancha y era capaz de almacenar miles de caracteres. El segundo fue creado por Petr Smirnov-Troyanskii y consistía de un diccionario que almacenaba las palabras en su forma base (el infinitivo de un verbo, el nominativo singular de un sustantivo, etc.). Era empleado en la segunda fase de un proceso de traducción , después de hacer el análisis del lenguaje origen se buscaban en la máquina las palabras en su forma base del idioma origen y se obtenían sus contrapartes del idioma destino en su forma base [Hutchins 1992, p.5].

1.8.1. Etapa pre-ALPAC

Al aparecer las computadoras después de la Segunda Guerra Mundial, se pensó que éstas podían ser empleadas en la traducción de idiomas. Como la traducción es una tarea relativamente fácil para los humanos, se creyó, que analizando el proceso de traducción empleado por los humanos, sería sencillo trasladarlo a las computadoras y hacer que estas tradujeran.

Warren Weaver y los primeros años

El punto que se ha considerado como el nacimiento de la traducción automática, tal y como se conceptualiza hoy en día, se llevó a cabo en 1947, cuando Warren Weaver, quien era vicepresidente de la fundación Rockefeller, escribió una carta a Norbert Wiener del Instituto Tecnológico de Massachusetts (MIT). En esa carta Weaver pregunta a Wiener acerca de la posibilidad de la traducción automática, a lo que Wiener demuestra poco interés. El 15 de Julio de 1949, Weaver realiza un documento en el cual sugería el concepto de traducción automática y lo distribuye a algunas personas que probablemente tuvieran algún interés en desarrollarlo. Este documento también sugería algunos métodos para llevar a cabo la traducción automática y contenía algunas cuestiones teóricas, como los múltiples significados de unidades lingüísticas, la base lógica del lenguaje, las técnicas de criptografía y la necesidad de un análisis del lenguaje. La distribución de este documento desencadenó la investigación en traducción automática en los Estados Unidos.

Un par de años más tarde existían ya un gran número de centros de investigación de traducción automática en los Estados Unidos, entre ellos el del Instituto de Tecnología de Massachusetts (MIT), la Universidad de Washington en Seattle, la Universidad de California en Los Ángeles (UCLA), la Corporación RAND y la Oficina Nacional de Estándares. En 1950, Erwin Reifler de la Universidad de Washington introduce los conceptos de pre-edición y post-edición. Yehoshua Bar-Hillel del MIT fue designado como el primer investigador de tiempo completo de traducción automática en 1951, y un año más tarde convocó a la primera reunión acerca de traducción automática donde se establecieron las pautas de investigación para el futuro. Uno de los logros de esta reunión fue el establecimiento de grupos de investigación en la Universidad de Hardvard y la Universidad de Georgetown.

De la investigación realizada en esta época surgieron algunos conceptos que son usados hoy en día, como el análisis morfológico y sintáctico, sugerencias de escribir textos en un lenguaje controlado, búsqueda automática en diccionarios, resolución de homografías, representaciones interlingua, argumentos para construir sistemas basados en un enfoque de sublenguaje y la necesidad de la intervención humana hasta que una traducción completamente automatizada fuera posible.

El sistema de Georgetown

La primera demostración pública de un sistema de traducción automática se realizó en 1954 en la Universidad de Georgetown, como resultado de la colaboración de esta universidad e IBM. El sistema traducía 49 frases tomadas de textos de química del ruso al inglés. El diccionario incluía un vocabulario muy restringido de 250 palabras y tan solo 6 reglas de gramática en ruso. Sus resultados fueron de una calidad aceptable, demostrando la posibilidad de la traducción automática. Esto condujo a una investigación a gran escala en Estados Unidos e inspiró para iniciar la investigación en otras partes del mundo como la Gran Bretaña y Rusia.

Los años de optimismo

En Octubre de 1956 se lleva a cabo la primera reunión internacional sobre traducción automática organizada por el MIT. Asistieron personas de Estados Unidos, Canadá y Gran Bretaña. Además se recibieron algunos escritos de Rusia. En esta reunión se mostró el curso de las investigaciones de los grupos hasta esa fecha establecidos.

Durante la segunda parte de la década de los 50's se vive una etapa de gran optimismo para la traducción automática. Se inicia la investigación en nuevos grupos en los Estados Unidos como los de Michigan, IBM, Texas, la Universidad de California en Berkeley y otros que no duraron mucho tiempo. En la URSS se inician proyectos en el instituto de Lingüística de Moscú y la Universidad de Leningrado. En Europa Occidental son iniciados proyectos en Italia (Universidad de Milán), Francia (Universidad de Grenoble) y Gran Bretaña (Unidad de Investigación del Lenguaje en Cambridge). El trabajo en estos grupos tenía diferentes enfoques. Algunos pretendían construir lo más pronto posible un sistema que funcionara, basados en enfoques estadísticos y de prueba y error, mientras que otros adaptaron un enfoque más teórico que trataba con investigación lingüística. Creían que primero era necesario llevar a cabo una investigación básica antes de contemplar el

desarrollo de un sistema que funcionara, esto dio como resultado contribuciones a la teoría lingüística. Otra de las diferencias entre los grupos de investigación consistía en la forma de traducción. Algunos grupos lo hacían palabra por palabra y usando información lexicográfica, mientras que los demás se concentraban en los problemas de sintaxis y de manipulación estructural. Los resultados obtenidos en esos años han sido importantes por mucho tiempo no sólo para la traducción automática sino también para la lingüística computacional y la inteligencia artificial. Una característica en común que se presentó en este periodo fue el adoptar la estrategia directa, aunque también surgen las primeras ideas de las estrategias interlingua y de transferencia. Pero el principal problema que tuvieron que enfrentar fue la poca capacidad de almacenamiento y velocidad de procesamiento de las computadoras de su época.

El Reporte de Bar-Hillel

Después del gran optimismo y de las predicciones de grandes avances, la complejidad de los problemas lingüísticos fue mas evidente y la calidad de las traducciones se mantenía en un nivel muy bajo. Fue entonces cuando Yehoshua Bar-Hillel publicó su crítica contemporánea de la traducción automática en 1960. Esta crítica estaba basada en la visita que había realizado a la mayoría de los centros de investigación de traducción automática en Estados Unidos y Gran Bretaña. Estableció que la meta de producir una Traducción Completamente Automatizada de Alta Calidad (FAHQT) no podría ser alcanzada, ni siquiera en textos de carácter científico. Bar-Hillel definió a la FAHQT como aquella que comprende a los sistemas que producirían resultados que no se podrían distinguir de los trabajos realizados por humanos. Decía que las barreras semánticas sólo podían ser resueltas a través de la inclusión de una gran cantidad de conocimiento del mundo real. Esto lo demostró a través de su ahora muy famoso ejemplo:

Little John was looking for his toy box. Finally, he found it. The box was in the pen. John was very happy [Bar-Hillel, 1960].

(El pequeño John estaba buscando su caja de juguetes. Finalmente, la encontró. La caja estaba en el corral. John estaba muy feliz.)

La palabra inglesa *pen* tiene al menos dos posibles significados, un instrumento de escritura y corral para jugar. Para un humano es muy sencillo resolver esta ambigüedad debido al conocimiento que posee. No es posible que una caja se encuentre dentro de un bolígrafo, pero para que una computadora pueda resolver esta ambigüedad debe

proporcionársele este conocimiento. Bar-Hillel recomendó que deberían de perseguirse metas menos ambiciosas que la FAHQT, es decir, sistemas ayudados mediante la intervención humana. Actualmente, muchos de los puntos argumentados por Bar-Hillel han podido ser resueltos con los avances en lingüística computacional y análisis semántico.

1.8.2. El reporte ALPAC

Debido a la posición que tenía Bar-Hillel dentro del mundo de la traducción automática, sus críticas fueron muy consideradas por las personas que proporcionaban los recursos para la investigación de la traducción automática en los Estados Unidos, quienes crearon en 1964 el Comité Consultivo de Procesamiento de Lenguaje en forma Automática (Automatic Language Processing Advisory Committee, ALPAC) para que informara del estado actual y de las perspectivas de la traducción automática. El comité realizó estudios sobre la demanda, oferta y costo de las traducciones, sobre la demanda y disponibilidad de traductores, hizo una evaluación sobre los resultados de la traducción automática y los costos de la post-edición.

El resultado fue una publicación ahora conocida como el reporte ALPAC. Este reporte fue ampliamente considerado como limitado, influenciado e inexacto [Hutchins 1992, p.7], aunque algunas de sus evaluaciones eran ciertas, como la que describía el estado de la traducción automática en esa época. El punto decisivo que desencadenó los efectos que causaría el reporte se dieron en las conclusiones. El comité concluyó que se debería poner más énfasis en el desarrollo de herramientas de ayuda para los traductores (como diccionarios mecanizados) que en la traducción automática misma, debido a que no había ningún sistema de traducción funcionando y no lo habría en un futuro inmediato. Se creía que la traducción automática había sido una falla muy costosa. La principal consecuencia de este reporte fue la reducción del apoyo económico a los proyectos de traducción automática, lo que casi acabo con la investigación en los Estados Unidos por más de una década. De los 10 grupos de investigación que existían en 1963, sólo quedaban tres en 1968. La influencia de este reporte fue más allá de los Estados Unidos, en Gran Bretaña desaparecieron los tres centros de investigación de traducción automática que existían, en la URSS y Japón la investigación continuó pero a niveles muy reducidos.

1.8.3. Etapa post-ALPAC

Después del reporte ALPAC, la investigación en traducción automática se realizó fuera de los Estados Unidos, principalmente en Europa y Canadá. Los investigadores en traducción automática ya no buscaban realizar trabajos que dieran traducciones estilísticamente perfectas, sino que fueran legibles y que concordaran con el original.

GETA

En Grenoble, Francia, durante 1960 se había creado el CETA (Centre d'Etudes de la Traduction Automatique). Este grupo de investigación intentó el desarrollo de un sistema interlingua que tradujera del ruso al francés, aunque también se realizaron investigaciones para traducir del alemán al francés y del japonés al francés. El sistema, llamado sistema CETA no dio los resultados esperados y fue suspendido en 1970. Para 1971, este grupo de investigación cambia su nombre por el de GETA (Groupe d'Etudes pour la Traduction Automatique) e inician el desarrollo del sistema con una estrategia de transferencia llamado Ariane o sistema GETA. El principal trabajo de investigación se hizo del ruso al francés, aunque también se hicieron investigaciones del alemán al francés. Este sistema es importante porque el trabajo realizado en Grenoble fue muy influyente en posteriores proyectos. Ariane tiene una fase de análisis y otra de generación divididas cada una en módulos morfológico y sintáctico. La parte de transferencia está dividida en una parte léxica y en otra estructural. El sistema es multilingüe debido a que los algoritmos y formalismos utilizados son independientes del idioma que se esté manejando. La investigación en este proyecto finalizó en 1987.

Susy

Hacia 1960, se estableció un grupo de investigación en Saarbrücken, Alemania. Para 1967 iniciaron el desarrollo de un sistema multilingüe llamado Susy (*S*aarbrücken *Ü*bersetzung*SY*stem, 'Sistema de traducción Saarbrücken'). La investigación se inició del ruso al alemán, posteriormente también se trabajó con el inglés, el francés y el Esperanto, aunque el mayor énfasis fue puesto en el análisis y generación del alemán. Susy es un sistema de transferencia con etapas de análisis y generación monolingüe y una etapa de transferencia bilingüe. Los textos de entrada pueden ser pre-editados opcionalmente. La etapa de análisis tiene ocho subprocesos entre los cuales se encuentran una etapa de análisis morfológico, de desambiguación de homografías y de desambiguación semántica. El sistema es altamente

modular y todos los módulos son realizados en una secuencia estricta. La calidad de las traducciones de Susy son comparables a las de cualquier otro sistema contemporáneo. En 1986, los trabajos en este sistema fueron terminados debido a que los investigadores en Saarbrücken se dedicaron al proyecto Eurotra (ver abajo).

Systran

El sistema de traducción automática Systran se inició en Alemania en 1964 con los trabajos del ruso al inglés de Peter Toma. En 1968, Toma se traslada a La Jolla, California donde Systran inicia haciendo traducciones para la fuerza aérea de los Estados Unidos. Posteriormente Systran fue usado por la NASA en la misión del Apolo-Soyus en 1975. Para el año siguiente la Comunidad Económica Europea compra una versión inglés-francés de Systran y decide usarlo como herramienta de traducción. En 1978 la Comunidad Europea adquiere la versión francés-inglés la cual se había iniciado el año pasado en La Jolla. Posteriormente, en la Comunidad europea se desarrollaron módulos adicionales para la traducción entre otros idiomas, como por ejemplo, inglés-alemán, inglés-italiano, etc. En 1981 se abre en Luxemburgo un servicio piloto de traducciones y desde esa fecha, la producción de traducciones de Systran se ha elevado considerablemente, hasta más de 400,000 páginas por año. Actualmente, existen varios pares de idiomas disponibles para traducción como: alemán-inglés, inglés-japonés, japonés-inglés, inglés-ruso, inglés-español, español-inglés, inglés-portugués, alemán-francés, alemán-italiano, alemán-español, etc. Además se encuentran en desarrollo otros pares de idiomas que incluyen algunos idiomas europeos como el holandés, noruego, sueco, danés y finlandés. Systran es un sistema con una estrategia de traducción de tipo directa, el sistema más exitoso con este tipo de estrategia. Una de sus características que lo ha hecho tan versátil es la modularidad, debido a que tiene etapas muy bien definidas. Actualmente, Systran es empleado por diversos organismos alrededor del mundo como la General Motors de Canadá, la compañía Dornier en Alemania, los ferrocarriles nacionales alemanes(Bundesbahn) y la compañía Xerox. El trabajo realizado por Systran solamente en la compañía Xerox es de mas de 60,000 páginas al año.

Eurotra

Debido a las obvias limitaciones de Systran, hacia 1978 la Comunidad Económica Europea inicia el proyecto Eurotra, un ambicioso proyecto que traería el resurgimiento de la traducción automática como una disciplina científica. Eurotra consiste en el desarrollo de un sistema de traducción automática de avanzado diseño capaz de realizar traducciones entre cada uno de los idiomas oficiales de la CEE: Alemán, Danés, Español, Francés, Griego, Holandés, Inglés, Italiano y Portugués. Eurotra posee grupos de investigación en todos los países miembros, reuniendo aproximadamente a 160 investigadores. La cede de los cuarteles centrales se encontraba en Luxemburgo. El sistema fue pensado para desarrollarse en tres fases. La primera fase fue para establecer un marco de trabajo y definir la teoría lingüística básica y las especificaciones de software. En esta etapa se decidió que se utilizaría una estrategia de traducción de transferencia, lo cual requiere nueve módulos de análisis, 72 módulos de transferencia y nueve módulos de generación. La segunda fase fue destinada para realizar el trabajo lingüístico necesario para un sistema prototipo pequeño con aproximadamente 2500 palabras por cada idioma y la tercera parte tiene como objetivo el desarrollo de un sistema para tratar con textos más complejos, trabajando con alrededor de 20,000 palabras por idioma y progresivamente con textos menos restringidos. Algunos autores señalan que Eurotra tiene un tratamiento semántico muy simple [Hutchins 1992, p.257], debido a que solamente vincula algunas características semánticas a los nodos de las estructuras sintácticas. Eurotra ha sido sin duda el proyecto más ambicioso, aunque su trabajo como tal finalizo en el año de 1990. Actualmente, sólo se realizan proyectos de investigación por separado.

Metal

Después del reporte ALPAC, uno de los pocos grupos de investigación que continuaron su trabajo en Estados Unidos fue el del Centro de Investigaciones Lingüísticas (LRC, Linguistics Research Center) en Austin, Texas, quienes, a partir de 1961, desarrollaron un sistema de traducción automática con una estrategia de transferencia del alemán al inglés llamado Metal (*Me*chanical *T*ranslator and *A*nalysis of *L*anguage, Traductor Mecánico y Análisis del Lenguaje). La característica de este grupo fue iniciar con una investigación lingüística fundamental, y a partir de 1970 inician el desarrollo del sistema con una estrategia interlingua. En 1978, la compañía alemana Siemens inicia el apoyo a este proyecto y para 1980 se convierte en el único aportador de recursos. El sistema cambia de emplear una estrategia de interlingua a una estrategia de transferencia. En 1989 aparece el

primer sistema Metal en el mercado para traducciones del alemán al inglés. Es un sistema para computadoras de tipo mainframe. La versión de alemán al inglés fue seguida por traductores que traducían los pares de idiomas holandés-francés, francés-holandés, alemán-español, alemán-francés y alemán-danés. Actualmente, la coordinación de la investigación en Metal es hecha en Munich, la investigación inglés-alemán es hecha en el LRC, la del francés-holandés es hecha en la universidad de Leuven en Bélgica, la del alemán-español en la universidad de Barcelona y la de alemán-danés en Kolding, Dinamarca. El sistema Metal esta diseñado para trabajar con documentación técnica, la cual generalmente está llena de diagramas y tablas. Metal es capaz de extraer el material para traducirlo y ponerlo de nuevo en su lugar en el idioma destino. El sistema entero está implementado en LISP. Hay una clara separación entre las operaciones de traducción y los datos que son llamados durante las operaciones. Estos datos comprenden información léxica de un solo lenguaje e información léxica bilingüe. La exactitud del sistema es de un 75%. Los principales campos de aplicación han sido el de comunicaciones y el de procesamiento de datos. Metal es usado en más de veinte compañías aparte de Siemens.

TAUM-Météo

En Canadá, a partir de 1968 se crea el grupo de investigación TAUM (Traduction Automatique de l'Université de Montréal). Ahí se realizaron varios sistemas experimentales, pero sólo uno llegó a tener una utilidad práctica, el sistema TAUM-Météo. El sistema fue iniciado en 1974 y estaba destinado para traducir reportes climatológicos emitidos por el Departamento Ambiental Canadiense. Los reportes debían ser traducidos del inglés al francés. El desarrollo del sistema tomó dos años. Un prototipo fue presentado en 1976 y el sistema inició su operación a partir de 1977. Desde entonces y hasta hoy TAUM-Météo se encarga de traducir reportes climatológicos para el uso de prensa y televisión. El sistema está construido con una estrategia de transferencia. Posee un léxico muy reducido de sólo 1500 palabras, de las cuales algunos cientos son nombres de lugares. Con este escaso léxico acerca de un tema en especial, el sistema no tiene que resolver muchas ambigüedades. Debido a su limitado campo de acción, Météo es el sistema de traducción automática en operación más exacto del mundo. Su exactitud esta por encima del 90%. Traduce alrededor de 37,000 palabras diarias. Su éxito consiste en que sólo hace una tarea, y la hace muy bien. Este sistema es el mejor ejemplo del uso del sublenguaje en un sistema de traducción automática.

Proyecto Mu

Los trabajos sobre traducción automática en Japón comenzaron dentro del proyecto japonés de quinta generación. En la universidad de Tokio se inicia alrededor de 1980 el proyecto Mu de traducción automática, encabezado por Makoto Nagao. Este proyecto pretendía construir un sistema para traducir documentos científicos y técnicos del japonés al inglés. El sistema era básicamente un sistema de transferencia, aunque cada uno de sus etapas había conservado componentes de sistemas de la primera generación. Este proyecto fue patrocinado por el gobierno y la industria. Este sistema fue evaluado logrando cerca del 50% de traducciones correctas.

Proyectos Holandeses

Para la década de los 80s, se inician investigaciones con otro tipos de sistema. Aparecen algunos proyectos basados en una estrategia de traducción interlingua. Entre ellos se encuentran los proyectos holandeses Rosetta y DLT.

El proyecto Rosetta se inició en 1980 en los Laboratorios de Investigación de Philips en Eindhoven, Holanda. En la primera fase se pretende trabajar en la lingüística esencial, en el marco computacional y en la construcción de un sistema que traduzca oraciones cortas para los pares de idiomas holandés-inglés, holandés-español, inglés-holandés y español-holandés. La segunda fase que inició en 1989 está dedicada a la construcción de un sistema más robusto y la construcción de un sistema prototipo para una aplicación real. Hasta el momento toda la investigación se ha basado en fundamentos teóricos y lingüísticos. Este sistema está basado en la gramática de Montague que esta especificada por una serie de expresiones con sus categorías gramaticales y un conjunto de reglas sintácticas que señalan como estas expresiones pueden combinarse para formar nuevas expresiones. El análisis del idioma fuente en Rosetta es independiente de la generación del idioma destino. Actualmente, el proyecto está en una etapa muy temprana para realizar una evaluación, pero es un proyecto que está experimentando con nuevos principios dentro de la traducción automática, lo cual es muy útil para enriquecerla.

El proyecto DLT (*D*istributed *L*anguage *T*ranslation) es un proyecto realizado un Utrecht, Holanda por una compañía de software llamada Buro voor Systemontwikkeling (BSO). DLT pretende ser un sistema con una estrategia de traducción interlingua que abarque la traducción de varios idiomas europeos, inicialmente el francés, inglés, alemán e italiano; que pueda ser usado en computadoras personales como herramienta en la

comunicación de literatura informativa, tales como manuales y reportes. La representación interlingua esta basada en una modificación del idioma Esperanto, por lo que muchas veces se piensa que la etapa de análisis y generación son sistemas de traducción por sí mismos, uno del idioma fuente al Esperanto y otro del Esperanto al idioma destino. La ventaja de usar el Esperanto como representación es que se trata de lenguaje natural que tiene una expresividad, riqueza y flexibilidad mayores que las representaciones interlingua ya construidas, provee un vocabulario basado en las raíces indoeuropeas más comunes, es regular, consistente, e independiente de otros lenguajes.

Investigación en E. U. después del ALPAC

En los Estados Unidos después del dañino efecto del reporte ALPAC, se reinician los trabajos en traducción automática. Se crean varios proyectos, la mayoría de ellos basados en técnicas de inteligencia artificial. El primer sistema de este tipo fue desarrollado por Yorick Wilks en la universidad de Stanford en 1973. Posteriormente, en la universidad de Colgate se desarrolló de 1983 a 1988 un sistema interlingua llamado Translator. Por su parte, en la universidad de Yale se han desarrollado dos experimentos basados en técnicas de representación de conocimiento. Estos sistemas han sido los antecesores de los que se desarrollan actualmente en la universidad Carnegie Mellon y en la universidad estatal de Nuevo México.

En la universidad Carnegie Mellon en Pittsburgh, Pennsylvania, se desarrolla el proyecto que emplea técnicas de inteligencia artificial más importante, conocido como traducción automática basada en el conocimiento (Knowledge-Based Machine Translation). El principio fundamental en que se basa este proyecto es que la traducción automática debe ir más allá de la información puramente lingüística, la traducción envuelve la comprensión del texto. No se trata de una comprensión total, tal como lo hacen los humanos utilizando mecanismos de inferencia y comprensión de información implícita y explícita, sino que se trata de una comprensión que sea suficiente para la traducción a varios idiomas. Esa comprensión se logra a través del conocimiento del mundo real almacenado en las bases de conocimiento[Nirenburg 1992, p.27]. Este sistema tiene una estrategia de traducción interlingua. El texto fuente es tratado por un conjunto de programas de análisis y, usando el conocimiento almacenado en el léxico y la gramática del lenguaje fuente, producen una expresión en un lenguaje que representa el significado del texto sin ambigüedades. Esta expresión es la representación interlingua. Posteriormente, las representaciones pasan a un conjunto de programas que realizan la generación, la cual usa el conocimiento heurístico del

léxico y las gramáticas del lenguaje destino. Algunas partes del sistema todavía se encuentran en desarrollo como las bases de conocimiento, los léxicos y algunas áreas de procesamiento lingüístico.

El proyecto en la universidad estatal de Nuevo México, llamado *Pangloss*, es un sistema de traducción automática basado en el conocimiento que combina una estrategia de traducción interlingua con otros enfoques en un sólo sistema. Involucra la traducción del inglés, el español y el japonés.

Quizá el punto más importante durante los últimos diez años es la aparición de sistemas de traducción automática comerciales, principalmente en Japón y los Estados Unidos. La mayoría de estos sistemas dependen grandemente de la post-edición para producir traducciones aceptables, aunque algunos incluyen opciones de pre-edición.

La nueva importancia de la traducción automática en todo el mundo se muestra por el éxito de las conferencias internacionales acerca del tema denominadas Summits. La primera se llevó a cabo en Hokone, Japón, en 1987; la segunda en Munich, Alemania, en 1989; la tercera en Washington, E.U., en 1991; la cuarta en Nancy, Francia; y la quinta en Luxemburgo, en 1995. Estas conferencias atrajeron a numerosos investigadores, usuarios, patrocinadores, instituciones gubernamentales, industrias y cuerpos multinacionales. Reflejan un nuevo optimismo basado en los avances en la tecnología de las computadoras (velocidad de procesamiento, capacidad de memoria, lenguajes de programación de alto nivel, ambientes de programación interactivos, arquitecturas de computadoras), y en lingüística computacional (técnicas de análisis y generación de textos tanto sintáctica como morfológica).

Para el futuro se espera lograr una mayor precisión en los sistemas de traducción automática, teniendo en cuenta que en la actualidad, los mayores obstáculos a vencer se encuentran en el área lingüística y no en la tecnológica.

II. METODOLOGÍA EMPLEADA EN ARD

II.1. Estrategias de Diseño de Ard

A continuación se presentan las estrategias de diseño adoptadas en el desarrollo del sistema de traducción automática *Ard*.

Primeramente diremos que *Ard* se trata de un sistema *bilingüe* y *unidireccional*, es decir, involucra la traducción de textos del idioma holandés al español. El objetivo de *Ard* es mostrar algunos de los conceptos fundamentales de la traducción automática, es por eso que para este trabajo de tesis sólo se ha desarrollado como un sistema bilingüe y unidireccional, aunque con los fundamentos que posee *Ard*, puede extenderse a un sistema bidireccional. Para lograrlo es necesario crear un módulo de análisis para el español, un módulo de generación para el holandés y los diccionarios necesarios para operar cada módulo. Actualmente el sistema tiene, debido a su función y alcance, un diccionario pequeño de aproximadamente 250 palabras base y una selección de alrededor de 100 reglas gramaticales.

Mediante el *lenguaje controlado Ard* hace uso de la intervención humana antes del proceso de traducción. Esto debe realizarse ya que se necesita que se adapten las textos que se van a traducir a estructuras que *Ard* puede manejar. Una vez introducido el texto, el sistema genera por sí mismo su traducción.

Estrategia de Traducción

La estrategia de traducción adoptada en *Ard* es la estrategia de transferencia. Se optó por esta estrategia para aprovechar las similitudes existentes entre el holandés y el español. Si se hubiera empleado la estrategia directa se dificultaría el proceso de cambio de las estructuras entre los idiomas involucrados en el proceso de traducción. La estrategia Interlingua tiene el inconveniente que necesita del desarrollo de una representación intermedia, lo cual requiere de un trabajo de análisis arduo que no está dentro de los alcances de este trabajo de tesis.

Como todo sistema de Transferencia *Art* posee una etapa de *entrada*, una etapa de *análisis*, una etapa de *transferencia*, una etapa de *generación* y una etapa de *salida*. Cada una de estas etapas fue desarrollada como un módulo, del mismo. La estructura y funcionamiento de cada uno de los módulos se presentará con detalle en el capítulo III.

II.2. Elementos Computacionales

Los elementos computacionales que se tuvieron que [illegible] para la implementación de *Art* son el lenguaje de programación [illegible] y el tipo de computadora [illegible].

Lenguaje de programación

El sistema fue desarrollado en el lenguaje de programación Prolog, [illegible] y compilado con Prolog [illegible].

[illegible]

Como todo sistema de Transferencia *Ard* posee una etapa de *entrada*, una etapa de *análisis*, una etapa de *transferencia*, una etapa de *generación* y una etapa de *salida*. Cada una de estas etapas fue desarrolladas como un módulo del sistema. La estructura y funcionamiento de cada uno de los módulos se presentará con detalle en el capitulo III.

II.2. Elementos Computacionales

Los elementos computacionales que se tuvieron que seleccionar para la implementación de *Ard* son el lenguaje de programación (software) y el tipo de arquitectura (hardware).

Lenguaje de programación

El sistema fue desarrollado en el lenguaje de programación Prolog, utilizando el compilador Arity Prolog V. 6.1 de 16 bits.

Prolog (**Pro**gramming in **Log**ic) fue creado en 1972 en la universidad de Marsella. Se trata de un lenguaje de tipo declarativo, es decir, basa su funcionamiento en hechos y reglas. Un hecho expresa información que se considera verdadera, mientras que una regla dice en que forma los hechos pueden ser inferidos a partir de otros. Una de las ventajas de los lenguajes declarativos sobre los lenguajes procedurales (C, Pascal, Basic, etc.) es que en los lenguajes declarativos, el programador sólo debe concentrarse en la lógica del programa. El programador diseña las soluciones del programa mientras que el lenguaje se encarga de las decisiones de control, que en Prolog se lleva a cabo a través del un motor de inferencia que hace uso del concepto de *backtracking*. Debido a estas diferencias Prolog es muy utilizado en el desarrollo de aplicaciones para Inteligencia Artificial y especialmente para el campo que nos ocupa, el Procesamiento del Lenguaje Natural.

El lenguaje Prolog fue escogido para desarrollar el sistema *Ard* por la facilidad que presenta en la implementación de las gramáticas de cláusula definida (ver II.3.).

El tamaño de las estructuras utilizadas en *Ard* es muy variable. No es posible establecer de antemano el tamaño de la estructura a utilizar porque no se conoce el tamaño, ni las características de la oración que se va a analizar. Es por eso que se deben crear estructuras de tipo dinámico para almacenar la información generada por cada uno de los

módulos del sistema de traducción. En este punto es donde Prolog tiene una ventaja adicional sobre otros lenguajes de programación debido a que crea las estructuras por si mismo, liberando al programador de esta tarea.

Arquitectura

La arquitectura IBM-PC fue seleccionada para realizar el desarrollo del sistema de traducción automática debido a que *Ard* no requería de grandes recursos de hardware para operar y dado que esta arquitectura nos proporciona los recursos necesarios, además de la facilidad de acceso a ella, la hicieron una opción viable. En esta arquitectura también se tiene la posibilidad de trabajar con las letras propias del español, lo que no es fácil hacerlo en otras arquitecturas.

II.3. Elementos Lingüísticos

De los enfoques y teorías lingüísticas expuestas en el capítulo anterior, se escogieron los siguientes elementos para ser aplicados al sistema experimental *Ard*. Aunque el modelo lingüístico en que está sustentado *Ard* se basa principalmente en la gramática de cláusula definida, también adopta elementos de otros formalismos.

Gramática de cláusula definida

Se ha escogido la utilización del formalismo de las gramáticas de cláusula definida para el sistema de traducción *Ard* porque puede ser fácilmente implementado en el lenguaje de programación Prolog debido a que este lenguaje está basado en cláusulas de Horn (ver pag. 14), una ventaja adicional es la posibilidad de crear árboles de estructura al realizar el análisis gramatical de una oración.

Este principio es usado en la etapa de análisis de *Ard*, durante la aplicación del parser o analizador.

Otros formalismos

Gramática de Estructura de Frase (ver pag. 12). Los principios de agrupación, de dominancia y precedencia que caracterizan a la gramática de estructura de frase son aplicados en la etapa de análisis de *Ard*.

Estos principios se ven reflejados en las reglas de la gramática utilizada en el parser de *Ard*. Por ejemplo en la regla:

$$S \rightarrow NP\ V$$

Donde:

S = oración
NP = frase nominal
V = verbo

los elementos NP y V son agrupados bajo el elemento S, de ahí se dice que S domina a NP y V. El principio de precedencia se establece porque NP precede a V en la regla.

Gramática de Valencia. El principio de la gramática de valencia que dice que el número de complementos requeridos por un verbo no cambia de idioma a idioma ha sido empleado dentro del proceso de traducción debido a que los verbos traducidos del holandés al español conservan su número de valencia a través de la traducción (ver pag. 15).

Representación de Características. La mayoría de los elementos léxicos dentro de *Ard* poseen características sintácticas y/o semánticas, que son usadas a lo largo del proceso de traducción. Estas características sirven de base para la construcción de estructuras mayores a través del principio de *unificación*. Las estructuras generadas obtienen sus características propias a partir de las características de los elementos que las componen.

Estas características son usadas para:

- Realizar verificaciones de tipo sintáctico (género, numero, persona) al generar los árboles sintácticos.
- Tomar decisiones acerca del la palabra a seleccionar en la transferencia léxica.
- Hacer ajustes de género a los sustantivos o de tiempo persona y número para los verbos en la etapa de generación.

II.4. Problemas Lingüísticos

Algunos de los problemas de carácter lingüístico que se presentaron durante el desarrollo del sistema que pudieron ser resueltos son los siguientes:

•Cambio en el orden de los adjetivos en una frase nominal. En el holandés los adjetivos preceden al sustantivo, pero en español generalmente se colocan después de él. *Ard* posee una regla de transferencia que se encarga de invertir el orden entre el sustantivo y los adjetivos en una frase nominal.

NP -> DET,ADJP,N : NP1 → DET1,N1,ADJP1

En un ejemplo práctico, si se tiene la oración en holandés:

Het	rode	huis
DET	ADJ	N

el sistema traducirá la oración utilizando la regla de transferencia anterior, dando como resultado:

La	casa	roja
DET	N	ADJ

•En las oraciones que están formadas con un verbo compuesto (de dos o más palabras) el holandés coloca las palabras que forman el verbo en dos posiciones, el auxiliar en la posición del verbo y el resto de las palabras que forman el verbo al final de la oración. El español por su parte, siempre coloca todas las palabras que forman el verbo en el mismo lugar en la oración. Para vencer este problema se tuvieron que crear algunas reglas de gramática que permitieran reconocer cuales son las palabras que forman el verbo dentro de una oración en holandés, para lo cual se creó un concepto llamado EVP (frase verbal al final de la oración). En el se agrupan los variedades que se presentan en el verbo al final de la oración. Las reglas de gramática usadas en la etapa de análisis para identificar el EVP son:

EVP → PART AUXINF
EVP → PART
EVP → MOD INF
EVP → AUXINF MOD INF
EVP → INF
EVP → PREF MOD REST
EVP → PREF AUXINF MOD REST
EVP → PREF

Donde:

PART. Pasado participio. Los participios de los tiempos perfectos se colocan al final de la oración en holandés. En la oración *Mijn vader heeft de auto gewassen* (mi papá ha lavado el auto) el participio *gewassen* está colocado al final de la oración.

INF. Verbo en infinitivo. El uso de un verbo modal o de los auxiliares para formar los tiempos futuro y condicional obligan al verbo principal a pasar al final de la oración. Por ejemplo

Morgen zal ik naar mexico gaan (Mañana iré a México)

El auxiliar para el tiempo futuro *zal* se coloca en la posición del verbo. Mientras que el verbo en infinitivo *gaan* (ir) se coloca al final de la oración.

PREF. Prefijo. Existen verbos holandeses que al ser conjugados en presente o en pasado son partidos de manera que en la posición del verbo queda la parte finita del verbo, mientras que el prefijo que antecedía a esta parte del verbo se coloca al final de la oración. Estos verbos son llamados *verbos separables*. El prefijo es representado por **PREF** y como es un elemento de verbo que se coloca al final de la oración se coloca como parte de una EVP.

Ejemplo:

We gaan 's morgens naar het park uit (salimos al parque en las mañanas)

El verbo separable *uitgaan* (salir) es partido en dos al ser conjugado en tiempo presente. La parte finita del verbo *gaat* que se coloca en la posición del verbo y el prefijo *uit* debe ser colocado al final de la oración.

REST (Parte finita del verbo separable). Cuando un verbo separable está acompañado de un verbo modal y además el verbo separable está conjugado en tiempo futuro o condicional, el auxiliar se coloca en la posición del verbo y el prefijo del verbo separable, el verbo modal y la parte finita del verbo separable se colocan al final de la oración. Esta parte finita del verbo separable al final de la oración se denomina REST.

Ejemplo:

Ik zal morgen uit moeten gaan (Tendré que salir mañana)

El auxiliar para el tiempo futuro *zal* se coloca en la posición del verbo. Al final de la oración se encuentran el prefijo del verbo separable *uit*, el verbo modal *moeten* y la parte finita del verbo separable *gaan*.

AUXINF. Auxiliar en infinitivo. El holandés pone al final de la oración un auxiliar en infinitivo cuando se presentan los tiempos de futuro perfecto o condicional perfecto.

MOD. Verbo modal. Cuando un verbo modal esta conjugado en un tiempo que requiere el uso de un auxiliar, el auxiliar se coloca en la posición del verbo y el modal junto con el verbo principal se colocan al final de la oración.

Una vez identificadas la partes que componen el verbo es posible realizar los cambios estructurales necesarios para generar una oración en español. Esto es hecho a través de la aplicación de algunas reglas de transferencia que varían dependiendo de el resto de los elementos en la oración. Pero de forma general podemos decir que la regla que se aplica es:

S -> NP,V,NP2,EVP : S1 -> NP1,V1,NP3

En esta regla las partes V y EVP que representan el verbo en holandés son reducidas a la parte V1 para representar el verbo en español.

•Otro alcance importante del sistema es la capacidad para tratar una cantidad variable de adjetivos o frases preposicionales. En cualquier idioma, la cantidad de adjetivos que pueden modificar a un sustantivo no tiene límite, así como las frases preposicionales que pueden aparecen en una oración. *Ard* es capaz de manejar una frase nominal con uno, dos, tres o más adjetivos (ADJ), agrupándolos todos en una frase llamada ADJP. Las reglas de gramática que permiten reconocer los adjetivos y agruparlos en una frase son:

ADJP -> ADV ADJP
ADJP -> ADJ ADJP
ADJP -> ADJ

Para reconocer varias frases preposicionales en la misma parte de la oración, *Ard* utiliza reglas de gramática basadas en las empleadas para crear la frase ADJP. Las siguientes reglas agrupan las frases preposicionales (PP) en un elemento denominado PPS que :

PPS -> PP PPS
PPS -> PP

El orden que siguen las reglas de ADJP y PPS es muy importante, si hubiera un cambio, no sería posible reconocer los adjetivos y las frases preposicionales.

Existen algunos puntos lingüísticos que no pudieron ser resueltos por *Ard*. Entre ellos se encuentra la traducción correcta de las preposiciones. Aunque existen algunas técnicas que ayudan a resolver algunas de las traducciones, todavía no existe una teoría que permita traducir las preposiciones de un idioma a otro correctamente. El problema consiste en el hecho de que las preposiciones tienen varios sentidos, los cuales generalmente son traducidos en preposiciones distintas en otros idiomas, esto se debe a que idiomas y culturas diferentes interpretan de manera diferente las relaciones entre entidades. Por ejemplo en la oración *la pelota esta en el piso* la preposición *en* es cambiada por la preposición *op* (literalmente *sobre*) al traducirla al holandés, dando como resultado la oración *de ball staat op de grond*.

Hay ocasiones en que *Ard* no obtiene la conjugación correcta para los verbos. En holandés como otros lenguas germanas, los verbos no se conjugan en los mismos tiempos que utilizan las lenguas romances como el español. El tiempo presente del holandés corresponde al presente del modo indicativo y al presente del modo subjuntivo en español. *Ard* no puede distinguir la diferencia y traduce estos verbos conjugados en presente de indicativo. Algo similar ocurre con el pretérito del holandés, el cual puede ser traducido como pretérito o como copretérito al español. En este caso, *Ard* traduce el verbo conjugado en pretérito. Para vencer estos problemas lingüísticos es necesario realizar un análisis más profundo de cada uno de ellos, tratando de encontrar los casos en que debe usarse la traducción adecuada del verbo en cada caso.

III. IMPLEMENTACIÓN

Este capítulo describe a grandes rasgos la implementación del sistema de traducción automática *Ard*.

El sistema lleva a cabo la traducción de oraciones del holandés al español. Su objetivo principal es mostrar algunos de los conceptos que gobiernan la traducción automática. *Ard* fue diseñado para traducir una oración a la vez, las capacidades para establecer relaciones entre oraciones para traducir un texto no le fueron implementadas.

Dentro del contexto en que se desarrolla *Ard*, no es posible que funcione como un traductor de todo el idioma holandés debido a dos razones primordiales:

•La generación de las gramáticas para el análisis, traducción y generación que requiere un sistema de traducción, de manera que abarquen todo el rango de posibilidades que posee un idioma, requiere de un gran trabajo muy especializado en lingüística, además de un conocimiento muy preciso acerca de los idiomas involucrados. Ambas situaciones se encuentran fuera del alcance de este trabajo de tesis.

•La arquitectura (IBM-PC) y el compilador (Arity/Prolog V. 6.1) en los que se desarrolló el sistema no son capaces de almacenar y manejar el lexicón y las reglas de gramática necesarias para realizar un sistema que tradujese un idioma completo.

El alcance del sistema es, por tanto, la traducción de algunas decenas de palabras y estructuras de oraciones de manera que logre cumplir el objetivo para el que fue desarrollado.

III.1 Estructura de Ard

Ard está compuesto por seis módulos, los cuales actúan en forma secuencial para generar las traducciones del usuario:

•Módulo Principal. Es el módulo encargado del control del sistema. Indica el orden en que se ejecutan los módulos dentro de *Ard*.

•Módulo de Entrada. En este módulo se obtienen del usuario la oración que será traducida por el sistema, se divide la oración en palabras y se verifica que las palabras que forman parte de la oración se encuentren dentro del lexicón del sistema.

•Módulo de Análisis. A partir de la salida del módulo de entrada obtiene mediante un analizador (parser) la estructura de la oración, la cual es representada mediante una estructura conocida como árbol de estructura de frase.

•Módulo de Transferencia. Tomando como base la estructura generada en el módulo de análisis, el módulo de transferencia hace las transformaciones léxicas y estructurales necesarias para generar un árbol de estructura de frase del idioma destino.

•Módulo de Generación. Crea la traducción de la oración escrita por el usuario en el módulo de entrada a partir del árbol de estructura de frase generado en el módulo de transferencia.

•Módulo de Salida. Simplemente muestra al usuario la oración que se creó en el módulo de generación o indica si hubo algún error en el proceso.

Todos los módulos de *Ard* fueron codificados en Arity Prolog v. 6.1. La descripción detallada de la implementación de cada uno de los módulos es presentada a continuación.

III.2. Módulo Principal

El módulo principal está basado en el predicado *main*. Es un predicado que consta de un ciclo repeat-fail que itera hasta que el usuario presione la tecla "Enter" en lugar de escribir una oración. Esta acción hace que el predicado *end* se cumpla y el sistema termine su ejecución.

```
main:-
        definitions,
        repeat,
                screen,
                get_sentence(LISTA_S),
                tmove(10,3),
                parser(LISTA_S,HOL_ARBOL),
                transf(HOL_ARBOL,ESP_ARBOL),
                gener(ESP_ARBOL,SAL),
                ifthen(LISTA_S\=[],
                ifthenelse(SAL\=[],
                write_trad(SAL),
                write_error)),
        end(LISTA_S),!,current_window(_,main).
```

El predicado *definitions* guarda las definiciones de las ventanas usadas en el sistema, Los predicados *get_sentence*, *parser*, *transf* y *gener* son los predicados que llaman a los módulos de entrada, análisis, transferencia y generación respectivamente. El módulo de Salida se compone los predicados *write_trad* y *write_error*.

El módulo principal también posee una serie de predicados de propósito general que son usados durante todo el sistema de traducción. Estos predicados se presentan a continuación:

```
append([],L,L):-!.
append([X|L1],L2,[X|L3]):-
        append(L1,L2,L3).

member(X,[X|_]):-!.
member(X,[_|T]):-
        member(X,T).

nth(1,[H|_],H):-!.
nth(N,[_|T],X):-
        N1 is N-1,
        nth(N1,T,X).
```

El predicado *append* une dos listas en una sola, el predicado *member* nos dice si el elemento X es parte de una lista, mientras que el predicado *nth* obtiene el enésimo elemento de la lista.

III.3 Módulo de Entrada

Este módulo establece la interface con el usuario para obtener la oración a traducir. El predicado *get_sentence* consta de un ciclo repeat-fail que verifica que todas las palabras de la oración se encuentren dentro del lexicón de *Ard* para que la oración pueda ser manejada por el sistema.

```
get_sentence(LISTA_S):-
        repeat,
                tmove(3,3),
                write('FRASE EN HOLANDES A TRADUCIR:'),
                tmove(22,25),
                write('PRESIONE ''ENTER'' PARA TERMINAR'),
                tmove(6,3),
                read_line(0,S),
                string_lower(S,S1),
                makes_list(S1,LISTA_S),
                check_lex(LISTA_S,NO_LEX),
                ifthen(NO_LEX\=[],no_lex_write(NO_LEX)),
        NO_LEX==[],!.
```

La oración se obtiene mediante el predicado *read_line*, que después es convertida a letras minúsculas a través de *string_lower*. El predicado *make_list* separa las palabras de la oración y las pone en una lista. Después verifica que las palabras de la oración se encuentren en el diccionario de *Ard*. Si alguna de ellas no es localizada, se manda un mensaje al usuario mostrándole las palabras que no fueron encontradas.

III.4. Módulo de Análisis

La función del módulo de análisis dentro de *Ard* es la de establecer la validez de las oraciones de entrada, así como la de crear una árbol de análisis para la oración por medio de un parser o analizador. Para llevar a cabo esta función, el módulo de análisis se sirve de dos elementos fundamentales: La gramática y el diccionario de holandés.

Gramática

La gramática del holandés utilizada en *Ard*, fue desarrollada pensando en mostrar algunas de las diferencias más grandes que tiene este idioma con el español. Esta compuesta por una serie de reglas que permiten establecer la validez de una oración. Se dice que una oración es válida cuando cumple con alguna de las reglas.

Diccionario

El diccionario empleado en este módulo contiene las palabras del idioma holandés que pueden ser traducidas por el sistema. Existen dos estructuras bajo las que se almacenan las palabras del diccionario.

La primera de ellas almacena los verbos utilizados en el análisis:

```
hol_lex_verb(ga,gaan,[s,1,pres,in]).
hol_lex_verb(gaat,gaan,[s,23,pres,in]).
hol_lex_verb(gaan,gaan,[p,123,pres,in]).
hol_lex_verb(gaan,gaan,[inf,in]).
hol_lex_verb(ging,gaan,[s,123,past,in]).
hol_lex_verb(gingen,gaan,[p,123,past,in]).
hol_lex_verb(gegaan,gaan,[part,z,in]).
```

El primer argumento elemento del predicado es el verbo finito, el segundo corresponde a la forma infinitiva del verbo y el tercero se trata de una lista con las características que representan la conjugación del verbo finito.

La segunda de las estructuras se utiliza para almacenar las entradas del diccionario para el resto de categorías gramaticales.

```
hol_lex_noun(jan,[c,s,prop,hum,male]).
hol_lex_adv(later,[posl]).
hol_lex_pron(deze,[c,b,3,subp]).
hol_lex_det(mijn,[a,b,nd]).
```

En ella el primer argumento del predicado corresponde al elemento léxico almacenado, mientras que el segundo muestra las características de ese elemento. El tipo de la categoría gramatical almacenada se encuentra en el nombre del predicado, por ejemplo el predicado *hol_lex_noun* almacena un sustantivo, el predicado *hol_lex_adv* almacena un adverbio, etc.

La representación de todas las categorías gramaticales en el diccionario y la descripción del significado de las características empleadas en cada una de ellas pueden ser consultadas en el Apéndice C.

Para mostrar la forma en que está implementado el módulo de análisis de *Ard*, se tomará una parte del código, algunas de reglas de la gramática y ejemplos del diccionario y se explicará paso a paso el análisis de una oración ejemplo (la totalidad de las reglas de gramática usadas por el parser se encuentran en el Apéndice B).

La oración que se tomará de ejemplo para observar el comportamiento del modulo de análisis es *Jan leest een boek* (Juan lee un libro). Esta oración necesita de las reglas de gramática para poder ser analizada:

```
S -> NP  V  NP

NP -> PN
NP -> DET  N

PN -> Jan
V -> leest
DET -> een
N -> boek
```

Estas reglas se presentan codificadas en la fracción de código utilizada para efectuar el análisis de la oración ejemplo, el cual se presenta a continuación:

```
parser(LISTA,ARBOL):-
        sentence(LISTA,ARBOL,[]),!.

parser(_,[]):-
        write('Error de Gram tica').

sentence(LS,[s,FEAT,[NP,V,NP2]],REST):-          /* S -> NP,V,NP */
        noun_phrase(LS,NP,REST1),
        verb(REST1,V,REST2),
        get_feat(V,V_FEAT),
        member(tr,V_FEAT),
        nth(3,V_FEAT,TIME),
        get_feat(NP,NP_FEAT),
        nth(2,NP_FEAT,NUM1),
        nth(1,V_FEAT,NUM2),
        get_number(NUM1,NUM2,NUM),
        nth(3,NP_FEAT,PER1),
        nth(2,V_FEAT,PER2),
        get_person(PER1,PER2,PER),
        FEAT = [NUM,PER,TIME,tr],
        noun_phrase(REST2,NP2,REST),!.

noun_phrase(LS,[np,NP_FEAT,[PN]],REST):-          /*  NP ->  PN  */
        proper_noun(LS,PN,REST),
        get_feat(PN,NP_FEAT).

noun_phrase(LS,[np,NP_FEAT,[DET,N]],REST):-          /*  NP -> DET,NOUN  */
        deter(LS,DET,REST1),
        noun(REST1,N,REST),
        get_feat(DET,FEAT1),
        get_feat(N,FEAT2),
        nth(1,FEAT1,GEN1),
        nth(1,FEAT2,GEN2),
        get_gender(GEN1,GEN2,GEN),
        nth(2,FEAT1,NUM1),
        nth(2,FEAT2,NUM2),
        get_number(NUM1,NUM2,NUM),
        tailnth(3,FEAT2,NP_FEAT1),
        insertnth(NUM,1,NP_FEAT1,NP_FEAT2),
        insertnth(GEN,1,NP_FEAT2,NP_FEAT).

proper_noun(LS,[pn,FEAT,W],REST):-
        word(LS,W,REST),!,
        hol_lex_noun(W,SIN_FEAT),
        member(prop,SIN_FEAT),
        add_redundant_features(SIN_FEAT,EXT_FEAT),
        insertnth(3,3,EXT_FEAT,FEAT).
```

```
det(det(LS,[det,FEAT,W],REST):-
        word(LS,W,REST),!,
        hol_lex_det(W,FEAT).

noun(LS,[n,FEAT,W],REST):-
        word(LS,W,REST),!,
        hol_lex_noun(W,SIN_FEAT),
        add_redundant_features(SIN_FEAT,EXT_FEAT),
        insertnth(3,S,EXT_FEAT,FEAT).

word([W|REST],W,REST).

get_feat([_,FEATURES|_],FEATURES).

get_gender(GEN1,GEN2,GEN):-
        case([ GEN1==GEN2 -> GEN=GEN1,
               GEN1==a -> GEN=GEN2,
               GEN2==a -> GEN=GEN1 | fail]).

get_number(NUM1,NUM2,NUM):-
        case([ NUM1==NUM2 -> NUM=NUM1,
               NUM1==b -> NUM=NUM2,
               NUM2==b -> NUM=NUM1 | fail]).

get_person(PER1,PER2,PER):-
        case([ PER1==PER2 -> PER=PER1,
               PER1==123 -> PER=PER2,
               PER2==123 -> PER=PER1,
               (PER1==12,(PER2==1;PER2==2)) -> PER=PER2,
               (PER2==12,(PER1==1;PER1==2)) -> PER=PER1,
               (PER1==13,(PER2==1;PER2==3)) -> PER=PER2,
               (PER2==13,(PER1==1;PER1==3)) -> PER=PER1,
               (PER1==23,(PER2==2;PER2==3)) -> PER=PER2,
               (PER2==23,(PER1==2;PER1==3)) -> PER=PER1 | fail]).
```

LÉXICO

```
hol_lex_noun(jan,[c,s,prop,hum,male]).
hol_lex_noun(boek,[n,s,inan,conc,cont,neut]).
hol_lex_det(een,[a,s]).
hol_lex_verb(leest,lezen,[s,23,pres,tr]).
```

El módulo de análisis inicia en el predicado *parser*, el cual genera un mensaje de error en caso de que la oración almacenada en la variable LISTA_S no se ajuste a las reglas de gramática almacenadas en el parser, de ser este el caso, no es generado ningún árbol de análisis

Estructura base

Cuando una regla de gramática es cumplida dentro del parser, el predicado genera una estructura que consiste en una lista de tres elementos. Por ejemplo, para la regla

$S \rightarrow NP\ V\ NP$

se genera la estructura

[s,FEAT,[NP,V,NP]]

en ella el primero los elementos de la lista corresponde al nombre del nodo del árbol, el cual identifica el símbolo no terminal de la parte izquierda de la regla de gramática. Las características asociadas a ese nodo se encuentran representadas como una lista en la segunda posición de la estructura, es de esta lista de donde se toman los elementos necesarios llevar a cabo la *unificación* o para realizar alguna selección de alternativas. El tercer elemento de la estructura almacena los elementos que forman parte de ese nodo. Este elemento pueden ser una lista de símbolos no terminales, que a su vez formarán otros nodos, o pueden ser un símbolo terminal.

La estructura descrita es la utilizada en todos los nodos de los árboles empleados durante el sistema de traducción.

Análisis

El predicado *sentence* inicia el análisis de la oración que va del punto inicial LS al punto final REST. El motor de inferencia de Prolog trata de completar los predicados de una manera en la que no pasa a la siguiente cláusula en un predicado hasta haber completado todos los elementos que forman la primera cláusula. A continuación se mostrarán fragmentos del debugger de Arity Prolog donde se muestra la forma en que el analizador va construyendo el árbol de acuerdo a este principio. Los números "_XXXX" dentro del listado indican una variable no instanciada (sin valor). Una llamada CALL ocurre cuando se llama por primera vez a un predicado, generalmente este tipo de llamadas contienen una o más variables no instanciadas. La llamada EXIT indica que el predicado se cumplió con éxito y muestra los datos obtenidos para las variables que no estaban instanciadas. Una llamada FAIL indica que el predicado falló y que utilizará el backtracking para buscar otra alternativa.

```
CALL: hol_par([jan,leest,een,boek],_0144)
CALL: sentence([jan,leest,een,boek],_0144,[])
CALL: noun_phrase([jan,leest,een,boek],_CC24,_CC4C)
CALL: proper_noun([jan,leest,een,boek],_CDAC,_CC4C)
CALL: word([jan,leest,een,boek],_CF14,_CC4C)
EXIT: word([jan,leest,een,boek],jan,[leest,een,boek])
CALL: hol_lex_noun(jan,_CF58)
EXIT: hol_lex_noun(jan,[c,s,prop,hum,male])
CALL: member(prop,[c,s,prop,hum,male])
EXIT: member(prop,[c,s,prop,hum,male])
CALL: add_redundant_features([c,s,prop,hum,male],_CF88)
EXIT: add_redundant_features([c,s,prop,hum,male],[c,s,prop,hum,anim,conc,cont,male])
CALL: insertnth(3,3,[c,s,prop,hum,anim,conc,cont,male],_0478)
EXIT: insertnth(3,3,[c,s,prop,hum,anim,conc,cont,male],[c,s,3,prop,hum,anim,conc,cont,male])
EXIT: proper_noun([jan,leest,een,boek],
                  pn,[c,s,3,prop,hum,anim,conc,cont,male],jan],[leest,een,boek])
CALL: get_feat([pn,[c,s,3,prop,hum,anim,conc,cont,male],jan],_03D0)
EXIT: get_feat([pn,[c,s,3,prop,hum,anim,conc,cont,male],jan],
               [c,s,3,prop,hum,anim,conc,cont,male])
```

```
EXIT(8): noun_phrase([jan,leest,een,boek],
                     [np,[c,s,3,prop,hum,anim,conc,cont,male],
                         [[pn,[c,s,3,prop,hum,anim,conc,cont,male],jan]]],
                     [leest,een,boek])
CALL: verb([leest,een,boek], _0538, _0540)
CALL: word([leest,een,boek], A1B4, C0A0)
EXIT: word([leest,een,boek],leest,[een,boek])
CALL: hol_lex_verb(leest, _A1C8, A1C0)
EXIT: hol_lex_verb(leest,lezen,[s,23,pres,tr])
EXIT: verb([leest,een,boek],[v,[s,23,pres,tr],lezen],een,boek])
CALL: get_feat([v,[s,23,pres,tr],lezen], 0494)
EXIT: get_feat([c,[s,23,pres,tr],lezen],[s,3,pres,tr])
CALL: member(tr,[s,23,pres,tr])
EXIT: member(tr,[s,23,pres,tr])
CALL: nth(3,[s,23,pres,tr], _03B8)
EXIT: nth(3,[s,23,pres,tr],pres)
CALL: get_feat([np,[c,s,3,prop,hum,anim,conc,cont,male],
                   [[pn,[c,s,3,prop,hum,anim,conc,cont,male],jan]]], 03D0)
EXIT: get_feat([np,[c,s,3,prop,hum,anim,conc,cont,male],
                     [[pn,[c,s,3,prop,hum,anim,conc,cont,male],jan]]],
                [c,s,3,prop,hum,anim,conc,cont,male])
CALL: nth(2,[c,s,3,prop,hum,anim,conc,cont,male], _03EC)
EXIT: nth(2,[c,s,3,prop,hum,anim,conc,cont,male],s)
CALL: nth(1,[s,23,pres,tr], 0408)
EXIT: nth(1,[s,23,pres,tr],s)
CALL: get_number(s,s, _0424)
EXIT: get_number(s,s,s)
CALL: nth(3,[c,s,3,prop,hum,anim,conc,cont,male], 0440)
EXIT: nth(3,[c,s,3,prop,hum,anim,conc,cont,male],3)
CALL: nth(2,[s,23,pres,tr], _045C)
EXIT: nth(2,[s,23,pres,tr],23)
CALL: get_person(3,23, _0478)
EXIT: get_person(3,23,3)
CALL: _0310 = [s,3,pres,tr]
EXIT: [s,3,pres,tr] = [s,3,pres,tr]
CALL: noun_phrase([een,boek], _0330,[])
CALL: proper_noun([een,boek], _7914,[])
FAIL: proper_noun([een,boek], _7914,[])
CALL: deter([een,boek], _7914, _793C)
CALL: word([een,boek], _7BBC, _793C)
EXIT: word([een,boek],een,[boek])
CALL: hol_lex_det(een, _7BB4)
EXIT: hol_lex_det(een,[a,s])
EXIT: deter([een,boek],[det,[a,s],een],[boek])
CALL: noun([boek], _791C,[])
CALL: word([boek], _9380,[])
EXIT: word([boek],boek,[])
CALL: hol_lex_noun(boek, _83C4)
EXIT: hol_lex_noun(boek,[n,s,inan,conc,cont,neut])
CALL: add_redundant_features([n,s,inan,conc,cont,neut], _83DC)
EXIT: add_redundant_features([n,s,inan,conc,cont,neut],[n,s,inan,conc,cont,neut])
CALL: insertnth(3,3,[n,s,inan,conc,cont,neut], 8378)
EXIT: insertnth(3,3,[n,s,inan,conc,cont,neut],[n,s,3,inan,conc,cont,neut])
EXIT: noun([boek],[n,[n,s,3,inan,conc,cont,neut],boek],[])
CALL: get_feat([det,[a,s],een], _06AC)
EXIT: get_feat([det,[a,s],een],[a,s])
CALL: get_feat([n,[n,s,3,inan,conc,cont,neut],boek], 06C4)
EXIT: get_feat([n,[n,s,3,inan,conc,cont,neut],boek],[n,s,3,inan,conc,cont,neut])
CALL: nth(1,[a,s], _06E0)
EXIT: nth(1,[a,s],a)
CALL: nth(1,[n,s,3,inan,conc,cont,neut], _06FC)
EXIT: nth(1,[n,s,3,inan,conc,cont,neut],n)
CALL: get_gender(a,n, _0718)
EXIT: get_gender(a,n,n)
CALL: nth(2,[a,s], _0734)
EXIT: nth(2,[a,s],s)
CALL: nth(2,[n,s,3,inan,conc,cont,neut], _0750)
EXIT: nth(2,[n,s,3,inan,conc,cont,neut],s)
CALL: get_number(s,s, _076C)
EXIT: get_number(s,s,s)
```

```
EXIT: noun_phrase([een,boek],
                  [np,[n,s,3,inan,conc,cont,neut],
                      [[det,[een,s]],[n,[boek,s,3,inan,conc,cont,neut],boek]]],[])
EXIT: sentence([ik,leest,een,boek],
               [s,[p,1,pres,ti],
                  [[np,[c,s,3,prop,hum,anim,conc,cont,male],
                      [[n,[ik,s,3],prop,hum,anim,conc,cont,male],[ik]]],
                   [v,[p,1,pres,ti],lezen],
                   [np,[n,s,3,inan,conc,cont,neut],
                       [[det,[een,s],een],
                        [n,[n,s,3,inan,conc,cont,neut],boek]]]]],[])
EXIT: hol_par([ik,leest,een,boek],
              [s,[p,1,pres,ti],
                 [[np,[c,s,3,prop,hum,anim,conc,cont,male],
                     [[pn,[c,s,3,prop,hum,anim,conc,cont,male],[ik]]],
                  [v,[p,1,pres,ti],lezen],
                  [np,[n,s,3,inan,conc,cont,neut],
                      [[det,[een,s],een],
                       [n,[n,s,3,inan,conc,cont,neut],boek]]]]],[])
```

Al finalizar la acción del parser se tiene una representación del árbol de estructura de la oración proporcionada por el usuario.

En la etapa de análisis, las palabras insertadas en los nodos del árbol que contienen un símbolo terminal son las palabras que aparecen en la oración proporcionada por el usuario con excepción de los verbos. Para ellos el parser obtiene del léxico la forma infinitiva del verbo la coloca en el nodo que representa el símbolo terminal.

III.5. Módulo de Transferencia

El objetivo del módulo de transferencia es convertir el árbol de estructura del idioma holandés generado por el módulo de análisis, en un árbol de estructura del español. Esta tarea es llevada a cabo mediante una serie de transformaciones léxicas y estructurales.

Este módulo requiere del uso de las reglas de transferencia y de un diccionario que proporcione las equivalencias léxicas.

Las reglas de transferencia establecen los cambios estructurales que hay que realizar al árbol de estructura en holandés para realizar su contraparte en español. No en todas las reglas se lleva a cabo un cambio estructural. Por ejemplo la regla de transferencia:

$$S \rightarrow NP,V,NP2 : S1 \rightarrow NP1,V1,NP3$$

no se lleva a cabo una transformación estructural sino sólo una transformación léxica. Sin embargo en la regla:

$$S \rightarrow NP,V,NP2,EVP : S1 \rightarrow NP1,V1,NP3$$

se lleva a cabo un cambio en su estructura. La frase verbal al final de la oración (EVP) desaparece al ser cambiada al español, debido a que en éste los verbos se colocan ya conjugados en una sola posición. El conjunto completo de las reglas de transferencia puede ser consultado en el Apéndice D.

Las transferencias léxicas se llevan a cabo de dos formas, la directa que se ejemplifica con predicados:

```
tr_lex_verb(schaatsen,patinar).
tr_lex_adv(vandaag,hoy).
tr_lex_prep(uit,de).
tr_lex_noun(geld,dinero):-!
```

y la transformación léxica que requiere de verificar algunas de las características de la palabra para obtener su traducción:

```
tr_lex_pron(zij,ellos,FEAT):-
        member(p,FEAT),!.
tr_lex_det(mijn,mi,FEAT):-
        member(s,FEAT),!.
```

Para mostrar el funcionamiento del módulo de transferencia continuaremos realizando el proceso de traducción de la oración *Jan leest een boek*. El código necesario para convertir el árbol de estructura en holandés, en un árbol de estructura en español se lista a continuación, en el están marcadas cada una de las reglas de transferencia utilizadas:

```
transf(HOL,ESP):-
        tr_s(HOL,ESP),!.

/* S -> NP,V,NP2 : S1 -> NP1,V1,NP3 */

tr_s([s,FEAT,[[np,NP_FEAT,NP],[v,V_FEAT,V],[np,NP_FEAT2,NP2]]],
     [s,FEAT,[[np,NP_FEAT,NP1],[v,V_FEAT,[V1]],[np,NP_FEAT2,NP3]]]):-
        tr_np(NP,NP1,NP_FEAT),
        tr_v(V,V1),
        tr_np(NP2,NP3,NP_FEAT2),!.

/* NP -> PN : NP1 -> PN1 */

tr_np([[pn,FEAT,PN]],[[pn,FEAT,PN1]],NP_FEAT):-
        tr_lex_noun(PN,PN1).

/* NP -> DET,N : NP1 -> DET1,N1 */

tr_np([[det,FEAT1,DET],[n,FEAT2,N]],
      [[det,FEAT1A,DET1],[n,FEAT2,N1]],NP_FEAT):-
        tr_lex_det(DET,DET1,NP_FEAT),
        che_gen_esp(DET,FEAT1,FEAT1A),
        tr_lex_noun(N,N1).

tr_v(V,V1):-
        tr_lex_verb(V,V1).

che_gen_esp(DET,FEAT,FEAT1):-
        member(DET,[de,het,'''t',een,deze,dit,die,dat]),
        append(FEAT,[sg],FEAT1),!.
che_gen_esp(_,FEAT,FEAT).
```

```
LÉXICO
tr_lex_noun(jan,juan):-!.
tr_lex_noun(boek,libro):-!.
tr_lex_det(een,un,MEAT):-!.
tr_lex_verb(lezen,leer).
```

El predicado *transf* es la entrada al módulo de transferencia de ahí se aplica una de las reglas de transferencia para convertir una oración a través del predicado *tr_s*. Así sucesivamente se llaman a los predicados necesarios para hacer la transferencia del resto de la oración. Cabe destacar que el acceso a los predicados en este módulo es directo, debido a que la estructura a transformar se encuentra en la definición de los predicados. El predicado *che_gen_lex*, agrega la característica "sg" a un determinante si la traducción de éste al español se ve afectada por el género del sustantivo en español.

La actividad del módulo para efectuar la transferencia de la oración ejemplo se presenta a continuación:

```
CALL: transf([s,[s,3,pres,tr],
                [[np,[c,s,3,prop,hum,anim,conc,cont,male],
                    [[pn,[c,s,3,prop,hum,anim,conc,cont,male],jan]]],
                [v,[s,23,pres,tr],lezen],
                [np,[n,s,3,inan,conc,cont,neut],
                    [[det,[a,s],een],
                    [n,[n,s,3,inan,conc,cont,neut],boek]]]],_015C)
CALL: tr_s([s,[s,3,pres,tr],
                [[np,[c,s,3,prop,hum,anim,conc,cont,male],
                    [[pn,[c,s,3,prop,hum,anim,conc,cont,male],jan]]],
                [v,[s,23,pres,tr],lezen],
                [np,[n,s,3,inan,conc,cont,neut],
                    [[det,[a,s],een],
                    [n,[n,s,3,inan,conc,cont,neut],boek]]]],_015C)
CALL: tr_np([[pn,[c,s,3,prop,hum,anim,conc,cont,male],jan]],_B970,
                [c,s,3,prop,hum,anim,conc,cont,male])
CALL: tr_lex_noun(jan,_BB50)
EXIT: tr_lex_noun(jan,juan)
EXIT: tr_np([[pn,[c,s,3,prop,hum,anim,conc,cont,male],jan]],
                [[pn,[c,s,3,prop,hum,anim,conc,cont,male],juan]],
                [c,s,3,prop,hum,anim,conc,cont,male]]
CALL: tr_v(lezen,_B998)
CALL: tr_lex_verb(lezen,_B998)
EXIT: tr_lex_verb(lezen,leer)
EXIT: tr_v(lezen,leer)
CALL: tr_np([[det,[a,s],een],[n,[n,s,3,inan,conc,cont,neut],boek]],_B9B0,
                [n,s,3,inan,conc,cont,neut]]
CALL: tr_lex_det(een,_C4E0,[n,s,3,inan,conc,cont,neut]]
EXIT: tr_lex_det(een,un,[n,s,3,inan,conc,cont,neut]]
CALL: che_gen_esp(een,[a,s],_C4D8)
CALL: member(een,[de,het,'''t',een,deze,dit,die,dat])
EXIT: member(een,[de,het,'''t',een,deze,dit,die,dat])
CALL: append([a,s],[sg],_C4D6)
EXIT: append([a,s],[sg],[a,s,sg])
EXIT: che_gen_esp(een,[a,s],[a,s,sg])
CALL: tr_lex_noun(boek,_C500)
EXIT: tr_lex_noun(boek,libro)
EXIT: tr_np([[det,[a,s],een],[n,[n,s,3,inan,conc,cont,neut],boek]],
                [[det,[a,s,sg],un],[n,[n,s,3,inan,conc,cont,neut],libro]],
                [n,s,3,inan,conc,cont,neut]]
EXIT: tr_s([s,[s,3,pres,tr],
                [[np,[c,s,3,prop,hum,anim,conc,cont,male],
                    [[pn,[c,s,3,prop,hum,anim,conc,cont,male],jan]]],
                [v,[s,23,pres,tr],lezen],
```

```
                [np,[n,s,3,inan,conc,cont,neut],
                        [[det,[a,s],een],
                        [n,[n,s,3,inan,conc,cont,neut],[boek]]]]],
              [s,[s,3,pres,tr],
                [[np,[c,s,3,prop,hum,anim,conc,cont,marie],
                        [[pn,[c,s,3,prop,hum,anim,conc,cont,marie],[jan]]],
                [v,[s,23,pres,tr],[houdt]],
                [np,[n,s,3,inan,conc,cont,neut],
                        [[det,[a,s],een],
                        [n,[n,s,3,inan,conc,cont,neut],[boek]]]]]]).
EXIT: tranef([s,[s,3,pres,tr],
                [[np,[c,s,3,prop,hum,anim,conc,cont,male],
                        [[pn,[c,s,3,prop,hum,anim,conc,cont,male],[jan]]],
                [v,[s,23,pres,tr],[lezen]],
                [np,[n,s,3,inan,conc,cont,neut],
                        [[det,[a,s],een],
                        [n,[n,s,3,inan,conc,cont,neut],[boek]]]]],
              [s,[s,3,pres,tr],
                [[np,[c,s,3,prop,hum,anim,conc,cont,male],
                        [[pn,[c,s,3,prop,hum,anim,conc,cont,male],[juan]]],
                [v,[s,23,pres,tr],[leer]],
                [np,[n,s,3,inan,conc,cont,neut],
                        [[det,[a,s],un],
                        [n,[n,s,3,inan,conc,cont,neut],[libro]]]]]])
```

El árbol de estructura en español que se obtiene del módulo de transferencia es pasado como entrada al módulo de generación.

III.6. Módulo de Generación

La función del módulo de generación es obtener la oración en español resultante del proceso de traducción a partir del árbol de estructura generado en el módulo de transferencia. Está basado en reglas que obtienen las cadenas de texto para cada nodo del árbol de estructura, por ejemplo la regla S → NP , V obtiene la cadena completa para una oración debido a que representa el nodo superior de un árbol de estructura. Su código dentro del módulo de generación es:

```
gen_s([s,FEAT,[NP,V]],SAL):-
        gen_np(NP,SAL1),
        gen_v(V,FEAT,SAL2),
        concat([SAL1,' ',SAL2],SAL),!.
```

aquí se obtiene las cadenas de texto parciales para la frase nominal y para el verbo, después estas cadenas son unidas en una sola, obteniéndose la cadena que representa la regla.

En los nodos de nivel más bajo en el árbol (las hojas) las cadenas de texto son generadas de una manera distinta. Aquí solo se obtiene el tercer elemento del nodo, se le aplican algunas transformaciones (si es necesario) y se convierte a una cadena. Por ejemplo, el código que obtiene un pronombre del nodo *prom* es el siguiente:

```
gen_sp([np,_,[[pron,_,SAL]]],SAL):-
        atom_string(SAL,SAL),!.
```

Las transformaciones que se efectúan a los elementos de los nodos inferiores para obtener la cadena de texto que los representan son principalmente:

•Selección del determinante adecuado al sustantivo contenido en las frases nominales. Esta acción es realizada en el predicado que obtiene la cadena de texto de los determinantes:

```
gen_det(FEAT,DET,N,SAL):-
        member(sg,FEAT),
        spa_lex_gen_n(N,S_FEAT),
        not(member(m,S_FEAT)),
        get_det_fem(DET,SAL),!.
```

Verifica si la característica "sg" fue insertada al determinante en el módulo de transferencia, de ser así, verifica el género del sustantivo "N" mediante el predicado *spa_lex_gen_n*:

```
spa_lex_gen_n(hermana,[f]):-!.
spa_lex_gen_n(trabajo,[m]):-!.
spa_lex_gen_n(palabra,[f]):-!.
```

el cual guarda el género del sustantivo dentro de la lista que contiene las características. Si el sustantivo es femenino obtiene el correspondiente femenino del determinante "DET" mediante el predicado *get_det_fem* y lo regresa como salida.

Conjugación del verbo o los verbos de la oración. Los verbos contenidos en los nodos del árbol de estructura son verbos en infinitivo. Para obtener su conjugación y presentarla en la oración final el módulo de generación utiliza el predicado *gen_v*, el cual va a buscar la conjugación adecuada dependiendo de las características que presente el verbo dentro del léxico a través del predicado *spa_verb*. En este predicado se guardan para cada uno de los verbos las conjugaciones de los tiempos manejados en *Ard*:

```
spa_verb(comprar,compro,s,1,pres):-!.
spa_verb(comprar,compras,s,2,pres):-!.
spa_verb(comprar,compra,s,3,pres):-!.
spa_verb(comprar,compramos,p,1,pres):-!.
spa_verb(comprar,compran,p,2,pres):-!.
spa_verb(comprar,compran,p,3,pres):-!.
spa_verb(comprar,'compré',s,1,past):-!.
spa_verb(comprar,compraste,s,2,past):-!.
spa_verb(comprar,'compró',s,3,past):-!.
spa_verb(comprar,compramos,p,1,past):-!.
spa_verb(comprar,compraron,p,2,past):-!.
spa_verb(comprar,compraron,p,3,past):-!.
spa_verb(comprar,'compraré,',s,1,fut):-!.
spa_verb(comprar,'comprarás',s,2,fut):-!.
spa_verb(comprar,'comprará',s,3,fut):-!.
spa_verb(comprar,compraremos,p,1,fut):-!.
spa_verb(comprar,'comprarán',p,2,fut):-!.
spa_verb(comprar,'comprarán',p,3,fut):-!.
spa_verb(comprar,'compraría',s,1,cond):-!.
spa_verb(comprar,'comprarías',s,2,cond):-!.
spa_verb(comprar,'compraría',s,3,cond):-!.
spa_verb(comprar,'compraríamos',p,1,cond):-!.
spa_verb(comprar,'comprarían',p,2,cond):-!.
spa_verb(comprar,'comprarían',p,3,cond):-!.
spa_verb(comprar,comprado,b,123,part):-!.
```

Los totalidad de los recursos léxicos utilizados en el módulo de generación están compuestos por los predicados: *spa_lex_gen_n, get_det_fem, spa_verb* y *get_adj_fem*. Este último obtiene los equivalentes femeninos de los sustantivos.

Para mostrar con más detalle el funcionamiento del módulo de generación se utilizará como ejemplo la oración *Jan leest een boek*. El código necesario para efectuar la generación de esta oración a partir del árbol de estructura obtenido en el módulo de transferencia es el siguiente:

```
gener([],[]):-!.

gener(ESP,SAL):-
        gen_s(ESP,SAL),!.

/* S -> NP,V,NP */
gen_s([s,FEAT,[NP,V,NP2]],SAL):-
        gen_np(NP,SAL1),
        gen_v(V,FEAT,SAL2),
        gen_np(NP2,SAL3),
        concat([SAL1,' ',SAL2,' ',SAL3],SAL),!.

gen_np([np,_,[[pn,_,SAL1]]],SAL):-
        atom_string(SAL1,SAL),!.

gen_np([np,_,[[det,FEAT1,DET],[n,_,N]]],SAL):-
        gen_det(FEAT1,DET,N,SAL1),
        concat([SAL1,' ',N],SAL),!.

gen_det(FEAT,DET,N,SAL):-
        member(sg,FEAT),
        spa_lex_gen_n(N,S_FEAT),
        not(member(m,S_FEAT)),
        get_det_fem(DET,SAL),!.

gen_det(_,DET,_,DET).
```

```
gen_v([v,V_FEAT,V],FEAT,GAL):-
        not(member([_,_,_],V)),
        nth(1,FEAT,NUM),
        nth(2,FEAT,PER),
        nth(3,FEAT,TIME),
        nth(1,V,Vl),
        spa_verb(Vl,GAL,NUM,PER,TIME),!.

spa_lex_gen_n(libro,[m]):-!.

spa_verb_2(leer,leo,s,1,pres):-!.
spa_verb_2(leer,lees,s,2,pres):-!.
spa_verb_2(leer,lee,s,3,pres):-!.
spa_verb_2(leer,leemos,p,1,pres):-!.
spa_verb_2(leer,laen,p,2,pres):-!.
spa_verb_2(leer,laen,p,3,pres):-!.
```

Los pasos que sigue el módulo de generación para obtener la cadena correspondiente a la traducción de la oración ejemplo se presentan a continuación

```
CALL:gener([s,[s,3,pres,tr],
                [[np,[c,s,3,prop,hum,anim,conc,cont,male],
                        [[pn,[c,s,3,prop,hum,anim,conc,cont,male],juan]]],
                [v,[s,23,pres,tr],[leer]],
                [np,[n,s,3,inan,conc,cont,neut],
                        [[det,[a,s,sg],un],[n,[n,s,3,inan,conc,cont,neut],libro]]]],_0174)
CALL: gen_s([s,[s,3,pres,tr],
                [[np,[c,s,3,prop,hum,anim,conc,cont,male],
                        [[pn,[c,s,3,prop,hum,anim,conc,cont,male],juan]]],
                [v,[s,23,pres,tr],[leer]],
                [np,[n,s,3,inan,conc,cont,neut],
                        [[det,[a,s,sg],un],[n,[n,s,3,inan,conc,cont,neut[,libro]]]]],_0174)
CALL: gen_np([np,[c,s,3,prop,hum,anim,conc,cont,male],
                [[pn,[c,s,3,prop,hum,anim,conc,cont,male],juan]]],_E978)
CALL: atom_string(juan,_E978)
EXIT: atom_string(juan,$juan$)
EXIT: gen_np([np,[c,s,3,prop,hum,anim,conc,cont,male],
                [[pn,[c,s,3,prop,hum,anim,conc,cont,male],juan]]],
                $juan$)
CALL: gen_v([v,[s,23,pres,tr],[leer]],[s,3,pres,tr],_05F0)
CALL: member([_0EE0,_0EE8,_0EF0],[leer])
CALL: member([_0EE0,_0EE8,_0EF0],[])
FAIL: member([_0EE0,_0EE8,_0EF0],[])
FAIL: member([_0EE0,_0EE9,_0EF0],[leer])
CALL: nth(1,[s,3,pres,tr],_0F10)
EXIT: nth(1,[s,3,pres,tr],s)
CALL: nth(2,[s,3,pres,tr],_0F2C)
EXIT: nth(2,[s,3,pres,tr],3)
CALL: nth(3,[s,3,pres,tr],_0F48)
EXIT: nth(3,[s,3,pres,tr],pres)
CALL: nth(1,[leer],_0F64)
EXIT: nth(1,[leer],leer)
CALL: spa_verb(leer,_05F0,s,3,pres)
EXIT: spa_verb(leer,lee,s,3,pres)
EXIT: gen_v([v,[s,23,pres,tr],[leer]],[s,3,pres,tr],lee)
CALL: gen_np([np,[n,s,3,inan,conc,cont,neut],
                [[det,[a,s,sg],un],[n,[n,s,3,inan,conc,cont,neut],libro]]],_0608)
CALL: gen_det([a,s,sg],un,libro,_5824)
CALL: member(sg,[a,s,sg])
EXIT: member(sg,[a,s,sg])
CALL: spa_lex_gen_n(libro,_59B4)
EXIT: spa_lex_gen_n(libro,[m])
CALL: member(m,[m])
EXIT: member(m,[m])
EXIT: gen_det([a,s,sg],un,libro,un)
CALL: concat([un,' ',libro],_0608)
EXIT: concat([un,' ',libro],$un libro$)
```

```
EXIT: gen_np([np,[n,s,3,[nan,conc,cont,neut]],
                    [[det,[a,s,sg],un],[n,[n,s,3,[nan,conc,cont,neut],libro]]],
              $un libro$)
CALL: concat([$juan$,' ',lee,' ',$un libro$], o[74])
EXIT: concat([$juan$,' ',lee,' ',$un libro$],$juan lee un libro$)
EXIT: gen_s([s,[c,3,pres,tr],
              [[np,[n,s,3,prop,hum,anim,conc,cont,male],
                    [[pn,[c,s,3,prop,hum,anim,conc,cont,male],juan]]],
              [v,[s,23,pres,tr],[lee]],
              [np,[n,s,3,[nan,conc,cont,neut],
                    [[det,[a,s,sg],un],[n,[n,s,3,[nan,conc,cont,neut],libro]]]]],
              $juan lee un libro$)
EXIT: gener([s,[s,3,pres,tr],
[[np,[c,s,3,prop,hum,anim,conc,cont,male],
[[pn,[c,s,3,prop,hum,anim,conc,cont,male],juan]]],
[v,[s,23,pres,tr],[lee]],
[np,[n,s,3,[nan,conc,cont,neut],
[[det,[a,s,sg],un],[n,[n,s,3,[nan,conc,cont,neut],libro]]]]],
$juan lee un libro$)
```

En la llamada Exit del predicado *gener* se obtiene la cadena de texto $juan lee un libro$, la cual es el resultado de el proceso de traducción.

III.7. Módulo de Salida

El módulo de salida presenta al usuario el resultado del proceso de traducción que puede ser la traducción de la oración dada por el usuario o un mensaje de error debido a algún problema surgido en el módulo de análisis. Esta constituido por los predicados:

```
write_trad(SAL):-
        tmove(10,3),
        write('TRADUCCION:'),
        tmove(12,3),
        write(SAL),nl,
        tmove(22,23),
        write('PRESIONE UNA TECLA PARA CONTINUAR'),
        wa(1,0),
        get0_noecho(CH),
        wa(1,15).

write_error:-
        tmove(22,23),
        write('PRESIONE UNA TECLA PARA CONTINUAR'),
        wa(1,0),
        get0_noecho(CH),
        wa(1,15).
```

write_trad despliega la traducción de la oración y *write_error* espera que el usuario presione una tecla después de presentarse un error en el módulo de análisis.

IV. OPERACIÓN DE ARD

En el presente capítulo se describirán los requerimientos y el funcionamiento del sistema de traducción automática *Ard*.

IV.1. Requerimientos del Sistema

Hardware y Software

Para poder funcionar correctamente el sistema *Ard* necesita de los siguientes elementos de Hardware y de Software. :

- Una computadora tipo IBM-PC con procesador 80286 o posterior.
- MS-DOS 2.0 o posterior.
- 520 Kbytes de memoria libre.
- Monitor EGA o superior

La falta de alguno de ellos haría que el sistema *Ard* sea inoperable.

Archivos de Ard

El sistema Ard se compone de los Archivos:

ARD.EXE , archivo ejecutable y

ARD.IDB , archivo que guarda la base de datos.

Ambos archivos deben encontrarse en el mismo directorio para que el sistema funcione.

IV.2. Manejo del Sistema

Para iniciar el sistema *Ard* es necesario teclear en el prompt de MS-DOS el comando *ard* y presionar la tecla "Enter":

C:\ard ↵

Una vez cargado el sistema, se muestra la pantalla de presentación:

```
UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
CAMPUS ARAGON

TESIS DE LICENCIATURA

"TRADUCCION AUTOMATICA Y ARD, UN TRADUCTOR HOLANDES-ESPAÑOL"

Por:
      Armin Ruelas Díaz
Asesor:
      Mtra. Ulrike Tallowitz Prado

                                                Agosto, 1995

PRESIONE UNA TECLA PARA CONTINUAR
```

Para continuar con la operación del sistema se presiona cualquier tecla, después de lo cual se presenta la pantalla principal:

```
                    ARD, TRADUCTOR HOLANDES-ESPAÑOL

FRASE EN HOLANDES A TRADUCIR:




                    PRESIONE 'ENTER' PARA TERMINAR
```

En ella *Ard* espera la oración en holandés a traducir. Se escribe la oración y se presiona la tecla Enter para traducirla.

Una vez proporcionada la oración el sistema realiza la traducción internamente y presenta el resultado en la pantalla:

```
ARD. TRADUCTOR HOLANDES-ESPAÑOL

FRASE EN HOLANDES A TRADUCIR:

morgen zal jan het boek lezen

TRADUCCION:
mañana juan leerá el libro

PRESIONE UNA TECLA PARA CONTINUAR
```

Presionando cualquier tecla el sistema regresa a la pantalla principal en donde espera otra oración para traducir.

Salida del sistema

Para salir del sistema, se debe presionar la tecla *Enter* en la pantalla principal, en lugar de escribir una oración para traducir.

Desplegado de errores

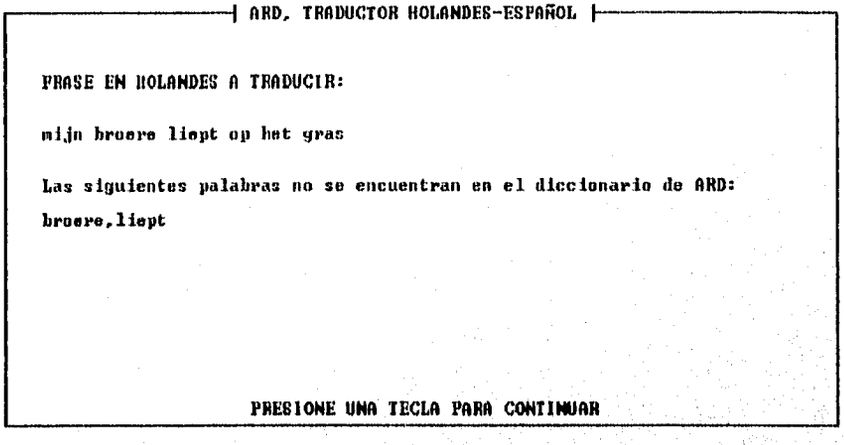
•En caso de que el sistema no disponga de la cantidad de memoria libre necesaria para poder ser ejecutado, se desplegará el error:

ERR 104 Not enough memory for startup

Una posible solución es liberar memoria quitando los programas residentes en memoria.

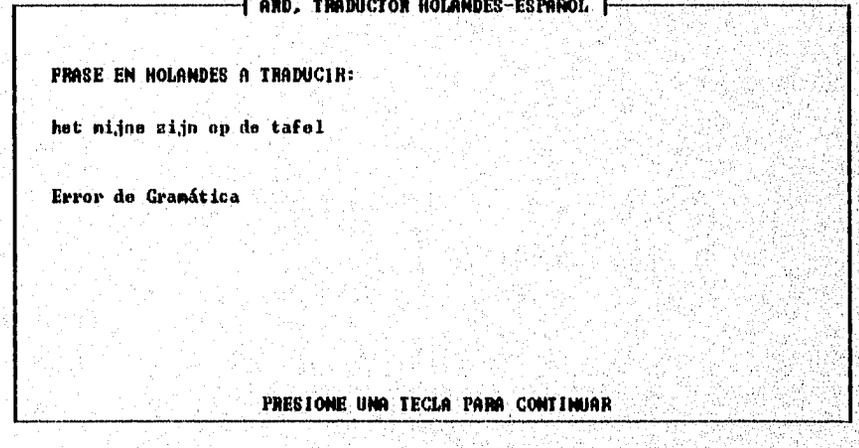
•El sistema *Ard* es incapaz de trabajar con palabras que no se encuentren dentro de sus diccionarios, es por eso que antes de efectuar una traducción revisa que todas las palabras proporcionadas se encuentren dentro de los diccionarios.

En caso cometer un error al escribir una palabra o proporcionar una palabra que no se encuentre en los diccionarios de *Ard*, el sistema manda un mensaje en el que presenta las palabras que no reconoció:



Al presionar cualquier tecla el sistema regresa a la pantalla principal.

•En caso de que *Ard* no pudiese analizar una oración, ya sea por que no conoce la estructura de la misma o porque es una oración anómala, el sistema manda un error a la pantalla mediante la frase "Error de Gramática".



Presionando cualquier tecla el sistema regresa a la pantalla principal.

Corrección de errores

En caso de presentarse un error de gramática o una palabra que el sistema no conozca y es posible corregir el error mediante la edición de la oración, las teclas del cursor pueden ayudar en la tarea. Las teclas "arriba" y "abajo" permiten desplegar las oraciones que han sido tecleadas durante la ejecución del sistema, de esta manera es posible desplegar la oración en la cual se cometió el error de escritura y corregirlo mediante las teclas "izquierda" y "derecha" que mueven el cursor a lo largo de la oración. El cursor se encuentra en el modo de sobreescritura, por lo que si se desea realizar la edición en el modo de inserción se debe presionar la tecla "Insertar". Una vez corregida la oración se presiona la tecla "Enter" para realizar su traducción.

V. CONCLUSIONES

En el presente trabajo se ha mostrado un panorama general de los conceptos que han sido desarrollados en el campo de la traducción automática. Este campo se basa en dos ciencias muy diferentes, la lingüística y la ciencia de la computación, por lo que se ha tratado de poner énfasis en ambas ramas para dar una mejor visión del área. El desarrollo de la parte lingüística requirió de una mayor investigación documental debido al poco conocimiento que se había adquirido acerca de esta ciencia.

El desarrollo del sistema de traducción automática holandés-español (*Ard*), cumple con el objetivo de presentar algunos de los conceptos en los que está basada la traducción automática. En él son empleados los concepto de la estrategia de transferencia para el desarrollo de un sistema de traducción, dividido en sus tres etapas principales análisis, transferencia y generación, el empleo de un parser en la etapa de análisis que construye un árbol de estructura basado en algunas gramáticas formales; el uso de reglas de transferencia léxica y estructural en la etapa de transferencia y el uso de diccionarios adecuados a cada una de las etapas del sistema.

Durante el desarrollo de *Ard* se resolvieron algunos de los problemas originados por las diferencias que existen entre los idiomas holandés y español, entre ellos se encuentran: el manejo de verbos compuestos dentro de una oración, el cambio en el orden que llevan los adjetivos y el sustantivo dentro de una frase nominal, y el reconocimiento de frases preposicionales y adjetivos contiguos (ver II.4).

No fue posible implementar un proceso para seleccionar la preposición adecuada dentro de las oraciones, debido principalmente a que no existe una teoría que proporcione resultados aceptables.

Originalmente se tenía pensado dotar a *Ard* de un léxico de varios cientos de palabras, pero durante su desarrollo el sistema se vio limitado por la cantidad de memoria que disponía en la arquitectura en la que se implementó. Es por eso *Ard* fue construido con un pequeño vocabulario de aproximadamente 250 palabras, de manera que le permitiera cumplir su objetivo. Para que un sistema sea medianamente eficiente para traducir textos es necesario que cuente con un vocabulario de 40,000 a 50,000 palabras.

Migrar el sistema a otra arquitectura o utilizar un compilador de 32-bits mejoraría el desempeño del sistema debido a que no tendría las limitaciones que posee ahora con los segmentos de memoria de 64K, permitiéndole contar con pilas de datos más grandes. Esto haría posible agregar una mayor cantidad de reglas a las gramáticas y de léxico a los diccionarios, con lo que se podrían traducir una mayor variedad de oraciones.

Dentro del ámbito lingüístico es posible continuar el desarrollo del sistema mediante la creación de nuevas reglas de gramática. Éstas podrían ser capaces de manejar nuevas estructuras de oraciones, o ayudar a resolver la ambigüedad que se presenta al intentar traducir los verbos en presente y pretérito del holandés al español (ver pag. 66).

Actualmente la investigación en traducción automática continúa, principalmente encaminada a resolver los problemas lingüísticos que han sido detectados. Incluso en la UNAM se llevan investigaciones al respecto con el desarrollo del sistema "cat2" que traduce textos entre el inglés y el español. La versión actual del sistema puede ser usada a través del Gopher.

Para el futuro, podemos esperar el desarrollo de teorías que permitan vencer algunos de los problemas que existen actualmente, y así generar traducciones de textos más exactas. No lograremos tener un sistema que traduzca toda clase de textos con la misma calidad que lo hacen los humanos, pero si podemos esperar el desarrollo de sistemas que auxiliados por medio de la pre-edición o la post-edición ayuden a agilizar las tareas de traducción de una manera confiable.

BIBLIOGRAFÍA

Allen, James. *Natural Language Understanding.* The Benjamin Cummings Publishing Company, Inc. California, USA. 1987.

Berlitz Spaans-Nederlads Woordenboek. Berlitz Publishing, S.A. Lausanne, Suiza. 1979.

Díaz Maeda, Alfredo Masanao. *Tesis de licenciatura: "Traductor sintáctico-semántico bidireccional inglés español basado en gramáticas de clausula definida y gramáticas de estructura de frase generalizada"* . UNAM. México. 1988.

Diccionario Universal Herder Español-Holandés Holandés-Español. 3a. Edición. Editorial Herder. Barcelona, España. 1989.

Dik, Simon C. *Functional Grammar in Prolog: An integrated Implementation for English, French and Dutch.* Mouton de Gruyter. Berlín, Alemania. 1992.

Firebaugh, Morris W. *Artificial Intelligence, A Knowledge-Based Approach.* PSW Kent Publishing Company. Boston, Massachusetts. 1989.

Gevarter, William B. *Intelligent Machines: An Introductory Perspective or Artificial Intelligence and Robotics.* Prentice Hall, Inc. New Jersey, USA. 1985.

Hartmann, R.R.K.; Stork, F.C. *Dictionary of Language and Linguistics.* Applied Science Publishers, Ltd. London. 1972.

Heniz-Dostert, Bozena; Mcdonald R. Ross; Zarechnak, Michael. *Machine Translation.* Mouto A Publishing. Berlín, Alemania. 1979.

Hirst, Graeme. Semantic Interpretation and The Resolution of Ambiguity. Cambridge University Press. Cambridge, Gran Bretaña. 1987.

Houët, Henriëtte. *Prisma Grammatica Nederlands.* Het Spectrum. Utrecht, Paises Bajos. 1988.

Hutchins, W. John. *Machine Translation: Past, Present and Future.* Ellis horwood. Chichester, Gran Bretaña. 1986.

Hutchins, W. John; Somers, Harold L. *An Introduction to Machine Translation.* Academic Press. San Diego, California. 1992.

Koolhoven, H. *Teaching Yourself Dutch, A Complete Course For Beginners*. NTC Publishing Group. Chicago, Illinois. 1994.

Kuiken, Folkert; Van Kalsbeek, Alice. *Code Nederlands*. Meulenhoff Educatief bv. Amsterdam, Paises Bajos. 1990.

Laffling, John. *Towards a High-Precision Machine Translation*. Foris Publications. Berlin, Alemania. 1991.

Marcus, Claudia. *Prolog Programming, Aplications for Database Systems, Expert Systems and Natural Language Systems*. Arity Corporation. E.U. 1986

Mayorcas, Pamel, Editor. *The Translation Environment 10 year on*. Aslib. Londres, Gran Bretaña. 1988.

Melbey, Alan. "On human-machine interaction in translation", *Machine translation: Theoretical and Methodological Issues*. Cambridge University Press. Cambridge, Gran Bretaña. 1987. pp. 145-154.

Nirenburg, Sergei, Editor. *Machine Translation: Theoretical and Methodological Issues*. Cambridge University Press. Cambridge, Gran Bretaña. 1987.

Nirenburg, Sergei; Carbonell, Jaime; Tomita, Masaru; Goodman, Kenneth. *Machine Translation: A Knowledge-Based Approach*. Morgan Kaufmann Publishers. San Mateo, California. 1992.

Reiner, Fernand G. *Colloquial Dutch*. Routledge. Londres, Inglaterra. 1981.

Slagter, Peter Jan; et al. *Van Dale Handwoordenboek Nederlands-Spaans*. Van Dale Lexicografie. Utrecht, Paises Bajos. 1993.

Slagter, Peter Jan; et al. *Van Dale Handwoordenboek Spaans-Nederlands*. Van Dale Lexicografie. Utrecht, Paises Bajos. 1993.

Stern, Henry R. 201 Dutch Verbs Fully Conjugated in All Tenses. Barron's Educational Series, Inc. Nueva York, USA, 1979.

Van Dam, C. F. A.; Oostendorp, H. TH. *Gramática Holandesa*. Consejo Superior de Investigaciones Científicas, Instituto "Miguel de Cervantes". Madrid, España. 1960.

Winston, Patrick Henry. *Artificial Intelligence*. 2a. Edición. Addison Wesley Publishing Company, USA, 1984.

APÉNDICE A

GLOSARIO

Adjetivo.- Palabra usada para calificar o describir un sustantivo.

Adjunto.- Palabra o frase que es usada para ampliar el significado de otra palabra o frase.

Adverbio.- Palabra que es usada para calificar un verbo, un adjetivo u otro adverbio.

Afijo.- Partícula añadida a la raíz de una palabra para formar una nueva.

Área de conocimiento.- Campo de estudio.

Ambiguo.- Palabra, frase u oración con más de una interpretación

Artículo.- Palabra usada como un adjunto de un sustantivo para modificar o limitar su significado.

Cabeza.- Palabra sintácticamente dominante en un grupo de palabras que puede tener la misma función sintáctica de todo el grupo si se presenta sola.

Categoría Gramatical.- Elementos gramaticales distinguidos por su forma, función o significado.

Cláusula.- Grupo de palabras que contiene un sujeto y un verbo.

Cláusula relativa.- Cláusula que se encuentra incluida dentro de la estructura de otra cláusula.

Conjunción.- Palabra usada para unir otras palabras o partes de oración.

Diccionario en línea: Programa que proporciona el significado o traducción de cualquier palabra o frase con sólo escribirla, de esta manera se ahorra la mayor parte del tiempo que es empleado en buscar palabras en diccionarios convencionales

Elipsis.- Es el fenómeno de no establecer explícitamente algunas palabras en una oración pero dejando el completarlas al lector o al escucha.

Expresión idiomática.- Grupo de palabras que tienen una connotación que usualmente no es igual a la suma de los significados individuales de las palabras.

Frase.- Grupo de palabras que forman una unidad sintáctica.

Frase nominal.- Palabra o grupo de palabras con un sustantivo o pronombre como cabeza. Dentro de una oración las frases nominales pueden funcionar como sujeto, objeto, o complemento.

Frase preposicional.- Grupo formado por una preposición y la frase nominal que le sigue

Gramática.- Conjunto de reglas que especifican las oraciones permitidas en un lenguaje.

Género.- Característica gramatical basada en la forma de los sustantivos, pronombres, adjetivos, artículos.

Homógrafo.- Aplicase a las palabras que se escriben de la misma forma pero que tienen un significado diferente.

Lenguaje natural.- Lengua nativa de un comunidad.

Lexicón. Conjunto de vocabulario o elementos léxicos de un lenguaje.

Morfema.- Unidad mínima de la gramática. Una palabra puede estar compuesta de uno o más morfemas.

Número.- Accidente gramatical por medio del cual los sustantivos adoptan formas diversas según se refieran a uno o más objetos.

Par de idiomas.- Son los idiomas involucrados en un proceso de traducción.

Polisémico.- Palabra con dos o más significados.

Prefijo.- Partícula antepuesta a ciertas palabras para modificar su significado.

Pragmática.- Es el estudio del lenguaje en el contexto.

Preposición.- Palabra usada junto con una frase nominal para mostrar la relación entre la frase y ptras palabras en la oración.

Pronombre.- Palabra empleada en lugar de un sustantivo.

Pronombre personal.- Pronombre que refiere a un a persona.

Pronombre posesivo.- Pronombre que designa la posición de un objeto por determinada persona gramatical

Raíz..- Morfema dentro de una palabra que tiene la información léxica principal.

Registro.- Variación en el lenguaje usado para un propósito específico. Estas variaciones generalmente son hechas en la formalidad (formal, coloquial), o en el tema tratado (terminología legal, de biología, de computación).

Sustantivo.- Palabra que denota una cosa, persona o concepto.

Semántica .- Es el estudio del significado del lenguaje.

Sintaxis.- Es el estudio del orden de las palabras en las oraciones y la manera en que se relacionan.

Sufijo.- Partícula añadida al final de las palabras para cambiar su significado

Verbo.- Palabras a las que pueden ser asignadas un numero, aspecto, persona, tiempo, modo y voz con base en el uso de inflexiones gramaticales o de palabras auxiliares.

Verbo auxiliar.- Palabra usada como adjunto con otro verbo. Generalmente modifican el tiempo de la conjugación del verbo.

APÉNDICE B

Gramática utilizada en el parser del idioma holandés

A continuación se presentan las reglas de gramática que fueron empleadas en el parser del idioma holandés. Los elementos entre paréntesis dentro de las reglas indican que el elemento es opcional, por lo tanto, una regla con un elemento opcional realmente describe dos reglas.

```
S -> NP  (PPS)
S -> NP  (PPS)  V  (PPS)
S -> NP  (PPS)  V  NP  (PPS)
S -> NP  (PPS)  V  (PPS)  EVP  (PPS)
S -> NP  (PPS)  V  NP  (PPS)  EVP  (PPS)
S -> (WVW)  V  NP  (PPS) ?
S -> (WVW)  V  NP  (PPS)  EVP  (PPS) ?
S -> (WVW)  V  NP  (PPS)  NP  (PPS) ?
S -> (WVW)  V  NP  (PPS)  NP  (PPS)  EVP  (PPS) ?
S -> APPS  V  NP  (PPS)
S -> APPS  V  NP  (PPS)  EVP  (PPS)
S -> APPS  V  NP  (PPS)  NP  (PPS)
S -> APPS  V  NP  (PPS)  NP  (PPS)  EVP  (PPS)

NP -> PRON
NP -> PN
NP -> DET  N
NP -> DET  ADJP  N
NP -> DET  POS_PRON

ADJP -> ADV  ADJP
ADJP -> ADJ  ADJP
ADJP -> ADJ

PPS -> PP  PPS
PPS -> PP

PP -> PREP  NP

EVP -> PART  AUXINF
EVP -> PART
EVP -> MOD  INF
EVP -> AUXINF  MOD  INF
EVP -> INF
EVP -> PREF  MOD  REST
EVP -> PREF  AUXINF  MOD  REST
EVP -> PREF

APPS -> ADV  PPS
APPS -> ADV
APPS -> PPS
```

Donde:

```
ADJ      = Adjetivo
ADJP     = Frase Adjetival
ADV      = Adverbio
ADVP     = Frase Adverbial
AUXINF   = Verbo auxiliar en infinitivo

DET      = Determinante

EVP      = Frase verbal al final de la oración

INF      = Verbo infinitivo

MOD      = Verbo modal

N        = Sustantivo
NP       = Frase nominal

PART     = Verbo en pasado participio
PN       = Nombre propio
POS_PRON = Pronombre posesivo
PP       = Frase Preposicional
PREF     = Prefijo de verbo separable
PREP     = Preposición
PRON     = Pronombre
PPS      = Frases Preposicionales

REST     = Parte finita del verbo separable

S        = Oración

V        = Verbo finito

WVW      = Palabra de inicio de pregunta
```

La implementación y la forma en que funcionan cada una de ellas es similar a la mostrada en el capítulo III. El objetivo de todas estas reglas es el de establecer la validez de una oración.

APÉNDICE C

Diccionario de holandés

El diccionario de holandés utilizado en el módulo de análisis se encuentran las palabras en holandés que pueden ser manejadas por el sistema. Se compone de una serie de predicados de dos argumentos que almacenan los sustantivos (*hol_lex_noun*), adjetivos (*hol_lex_adj*), determinantes (*hol_lex_det*), pronombres (*hol_lex_pron*), pronombres posesivos (*hol_lex_pos_pron*), preposiciones (*hol_lex_prep*), palabras para hacer preguntas (hol_lex_wvw) y adverbios (*hol_lex_adv*). En ellos, el primer argumento corresponde a la palabra almacenada y en el segundo almacenan una lista de características sintácticas y/o semánticas.

```
hol_lex_noun(jan,[c,s,prop,hum,male]).
hol_lex_noun(mexico,[n,s,prop,place,mass,neut]).
hol_lex_noun(boek,[n,s,inan,conc,cont,neut]).
hol_lex_noun(boeken,[n,p,inan,conc,cont,neut]).
hol_lex_noun(kat,[c,s,anim,male]).
hol_lex_noun(katten,[c,p,anim,male]).
hol_lex_noun(tafel,[c,s,inan,conc,cont,male]).
hol_lex_noun(tafels,[c,p,inan,conc,cont,male]).
hol_lex_noun(meisje,[n,s,hum,fema]).
hol_lex_noun(meisjes,[n,p,hum,fema]).
hol_lex_noun(brood,[n,s,inan,meal,cont,male]).
hol_lex_noun(broden,[n,p,inan,meal,cont,male]).

hol_lex_adj(leuk,[]).
hol_lex_adj(leuke,[decl]).
hol_lex_adj(groot,[]).
hol_lex_adj(grote,[decl]).

hol_lex_det(de,[c,b,da]).
hol_lex_det(de,[n,p,da]).
hol_lex_det(het,[n,s,da,nd]).
hol_lex_det(een,[a,s]).
hol_lex_det(mijn,[a,b,nd]).

hol_lex_pron(ik,[c,s,1,hum,subp]).
hol_lex_pron(ons,[c,p,1,hum,objp]).

hol_lex_prep(tussen,[]).
hol_lex_prep(in,[]).

hol_lex_adv(minder,[inten]).
hol_lex_adv(morgen,[pos1]).
hol_lex_adv(goed,[]).

hol_lex_wvw(welke,[wn]).
hol_lex_wvw(wie,[wv]).
hol_lex_wvw(hoeveel,[wv,wn]).
```

Los verbos por su parte son almacenados en un predicado de tres argumentos (*hol lex verb*), en donde el primer argumento es el verbo finito, el segundo el verbo en infinitivo y el tercero es una lista con de características sintácticas del verbo.

```
hol_lex_verb(denk,denken,[s,1,pres,in,tr]).
hol_lex_verb(denkt,denken,[s,2s,pres,in,tr]).
hol_lex_verb(denken,denken,[p,i3,pres,in,tr]).
hol_lex_verb(denken,denken,[inf,in,tr]).
hol_lex_verb(dacht,denken,[s,123,past,in,tr]).
hol_lex_verb(dachten,denken,[p,123,past,in,tr]).
hol_lex_verb(gedacht,denken,[part,b,in,tr]).

hol_lex_verb(doe,doen,[s,1,pres,in,tr]).
hol_lex_verb(doet,doen,[s,23,pres,in,tr]).
hol_lex_verb(doen,doen,[p,123,pres,in,tr]).
hol_lex_verb(doen,doen,[inf,in,tr]).
hol_lex_verb(deed,doen,[s,123,past,in,tr]).
hol_lex_verb(deden,doen,[p,123,past,in,tr]).
hol_lex_verb(gedaan,doen,[part,h,in,tr]).

hol_lex_verb(ga,gaan,[s,1,pres,in]).
hol_lex_verb(gaat,gaan,[s,23,pres,in]).
hol_lex_verb(gaan,gaan,[p,123,pres,in]).
hol_lex_verb(gaan,gaan,[inf,in]).
hol_lex_verb(ging,gaan,[s,123,past,in]).
hol_lex_verb(gingen,gaan,[p,123,past,in]).
hol_lex_verb(gegaan,gaan,[part,z,in]).
```

Las características que aparecen en las entradas del diccionario son compartidas por varías categorías gramaticales, es por eso que serán descritas juntas.

Características de género

c - común (género utilizado para referirse a los sust. que usan el artículo "de")

n - neutro (género utilizado para referirse a los sust. que utilizan el artículo "het")

a - representa ambos géneros, es utilizado en los determinantes que pueden ser agrupados con sustantivos de ambos géneros

Características de número.

s - singular

p - plural

b - palabras que pueden ser usadas en ambos casos

Características de persona

1 - primera persona

2 - segunda persona

3 - tercera persona

123 - palabra utilizada indistintamente para la primera, segunda y tercera persona.

23 - palabra utilizada indistintamente para la segunda y tercera persona

Características exclusivas de los determinantes

da - artículo definido

nd - determinantes que declinan los adjetivos cuando el sustantivo de la frase nominal es neutro y singular

Características semánticas (sustantivos)

anim - animado
inan - inanimado
conc - concreto
abst - abstracto
hum - humano
meal - alimento
place - lugar
time - tiempo
mass - incontable
cont - contable
prop - propio

Características exclusivas de los adjetivos

decl - declinado

Características exclusivas de los verbos

pres - presente
past - pretérito
part - pasado participio
z - verbos que forman los tiempos perfectos con el auxiliar zijn
h - verbos que forman los tiempos perfectos con el auxiliar hebben
tr - transitivo
in - intransitivo
mod - verbo modal
paux - verbos auxiliares para formar los tiempos perfectos
fut - futuro
cond - condicional

APÉNDICE D

Reglas de transferencia usadas en Ard

Las reglas de transferencia implementadas en *Ard* son usadas para realizar los cambios estructurales y léxicos a las oraciones durante el módulo de transferencia. Algunas de las reglas realizan un cambio tanto en la estructura de la oración como en sus elementos léxicos, mientras que otras sólo lo hacen en la parte léxica porque las estructuras tanto del español como del holandés son similares. A continuación se presenta el conjunto de reglas de transferencia que utiliza *Ard* excepto aquellas que realizan sólo las transferencias léxicas.

```
S -> NP : S1 -> NP1
S -> NP,PPS : S1 -> NP1,PPS1
S -> NP,V : S1 -> NP1,V1
S -> NP,PPS,V : S1 -> NP1,PPS1,V1
S -> NP,V,PPS : S1 -> NP1,V1,PPS1
S -> NP,PPS,V,PPS2 : S1 -> NP1,PPS1,V1,PPS3

S -> NP,V,NP2 : S1 -> NP1,V1,NP3
S -> NP,V,NP2,PPS : S1 -> NP1,V1,NP3,PPS1
S -> NP,PPS,V,NP2 : S1 -> NP1,PPS1,V1,NP3
S -> NP,PPS,V,NP2,PPS2 : S1 -> NP1,PPS1,V1,NP3,PPS3

S -> NP,V,EVP : S1 -> NP1,V1
S -> NP,V,EVP,PPS : S1 -> NP1,V1,PPS1
S -> NP,PPS,V,EVP : S1 -> NP1,PPS1,V1
S -> NP,PPS,V,EVP,PPS2 : S1 -> NP1,PPS1,V1,PPS3
S -> NP,V,PPS,EVP : S1 -> NP1,V1,PPS1
S -> NP,V,PPS,EVP,PPS2 : S1 -> NP1,V1,PPS4
S -> NP,PPS,V,PPS2,EVP : S1 -> NP1,PPS1,V1,PPS3
S -> NP,PPS,V,PPS2,EVP,PPS4 : S1 -> NP1,PPS1,V1,PPS6

S -> NP,V,NP2,EVP : S1 -> NP1,V1,NP3
S -> NP,V,NP2,EVP,PPS : S1 -> NP1,V1,NP3,PPS1
S -> NP,PPS,V,NP2,EVP : S1 -> NP1,PPS1,V1,NP3
S -> NP,PPS,V,NP2,EVP,PPS2 : S1 -> NP1,PPS1,V1,NP3,PPS3
S -> NP,V,NP2,PPS,EVP : S1 -> NP1,V1,NP2,PPS1
S -> NP,V,NP2,PPS,EVP,PPS2 : S1 -> NP1,V1,NP3,PPS4
S -> NP,PPS,V,NP2,PPS2,EVP : S1 -> NP1,PPS1,V1,NP3,PPS3
S -> NP,PPS,V,NP2,PPS2,EVP,PPS4 : S1 -> NP1,PPS1,V1,NP3,PPS6

S -> V,NP,? : S1 -> ¿,V1,NP1,?
S -> V,NP,PPS,? : S1 ->¿, V1,NP1,PPS1,?
S -> V,NP,EVP,? : S1 -> ¿,V1,NP1,?
S -> V,NP,EVP,PPS,? : S1 -> ¿,V1,NP1,PPS1,?
S -> V,NP,PPS,EVP,? : S1 -> ¿,V1,NP1,PPS1,?
S -> V,NP,PPS,EVP,PPS2,? : S1 -> ¿,V1,NP1,PPS4,?

S -> V,NP,NP2,? : S1 -> ¿,V1,NP1,NP3,?
S -> V,NP,NP2,PPS,? : S1 -> ¿,V1,NP1,NP3,PPS1,?
S -> V,NP,PPS,NP2,? : S1 -> ¿,V1,NP1,PPS1,NP3,?
S -> V,NP,PPS,NP2,PPS2,? : S1 -> ¿,V,NP1,PPS1,NP3,PPS3,?
```

```
S -> V,NP,NP2,EVP,? : S1 -> ¿,V1,NP1,NP3,?
S -> V,NP,NP2,EVP,PPS,? : S1 -> ¿,V1,NP1,NP3,PPS1,?
S -> V,NP,PPS,NP2,EVP,? : S1 -> ¿,V1,NP1,PPS1,NP3,?
S -> V,NP,PPS,NP2,EVP,PPS2,? : S1 -> ¿,V1,NP1,PPS1,NP3,PPS2,?
S -> V,NP,NP2,PPS,EVP,? : S1 -> ¿,V1,NP1,NP3,PPS1,?
S -> V,NP,NP2,PPS,EVP,PPS2,? : S1 -> ¿,V1,NP1,NP3,PPS4,?
S -> V,NP,PPS,NP2,PPS2,EVP,? : S1 -> ¿,V1,NP1,PPS1,NP3,PPS4,?
S -> V,NP,PPS,NP2,PPS2,EVP,PPS4,? : S1 -> ¿,V1,NP1,PPS1,NP3,PPS6,?

S -> WWW,V,NP,? : S1 -> ¿,V1,V1,NP1,?
S -> WWW,V,NP,PPS,? : S1 -> ¿,V1,V1,NP1,PPS1,?
S -> WWW,V,NP,EVP,? : S1 -> ¿,V1,V1,NP1,?
S -> WWW,V,NP,EVP,PPS,? : S1 -> ¿,V1,V1,NP1,PPS1,?
S -> WWW,V,NP,PPS,EVP,? : S1 -> ¿,V1,V1,NP1,PPS1,?
S -> WWW,V,NP,PPS,EVP,PPS2,? : S1 -> ¿,V1,V1,NP1,PPS4,?

S -> WWW,V,NP,NP2,? : S1 -> ¿,V1,V1,NP1,NP3,?
S -> WWW,V,NP,NP2,PPS,? : S1 -> ¿,V1,V1,NP1,NP3,PPS1,?
S -> WWW,V,NP,PPS,NP2,? : S1 -> ¿,V1,V1,NP1,PPS1,NP3,?
S -> WWW,V,NP,PPS,NP2,PPS2,? : S1 -> ¿,V1,V,NP1,PPS1,NP3,PPS2,?
S -> WWW,V,NP,NP2,EVP,? : S1 -> ¿,V1,V1,NP1,NP3,?
S -> WWW,V,NP,NP2,EVP,PPS,? : S1 -> ¿,V1,V1,NP1,NP3,PPS1,?
S -> WWW,V,NP,PPS,NP2,EVP,? : S1 -> ¿,V1,V1,NP1,PPS1,NP3,?
S -> WWW,V,NP,PPS,NP2,EVP,PPS2,? : S1 -> ¿,V1,V1,NP1,PPS1,NP3,PPS3,?
S -> WWW,V,NP,NP2,PPS,EVP,? : S1 -> ¿,V1,V1,NP1,NP2,PPS1,?
S -> WWW,V,NP,NP2,PPS,EVP,PPS2,? : S1 -> ¿,V1,V1,NP1,NP3,PPS4,?
S -> WWW,V,NP,PPS,NP2,PPS2,EVP,? : S1 -> ¿,V1,V1,NP1,PPS1,NP3,PPS3,?
S -> WWW,V,NP,PPS,NP2,PPS2,EVP,PPS4,? : S1 -> ¿,V1,V1,NP1,PPS1,NP3,PPS6,?

S -> APPS,V,NP : S1 -> APPS1,NP1,V1
S -> APPS,V,NP,PPS : S1 -> APPS1,NP1,V1,PPS1
S -> APPS,V,NP,EVP : S1 -> APPS1,NP1,V1
S -> APPS,V,NP,EVP,PPS : S1 -> APPS1,NP1,V1,PPS1
S -> APPS,V,NP,PPS,EVP : S1 -> APPS1,NP1,PPS1,V1
S -> APPS,V,NP,PPS,EVP,PPS2 : S1 -> APPS1,NP1,PPS1,V1,PPS3

S -> APPS,V,NP,NP2 : S1 -> APPS1,NP1,V1,NP3
S -> APPS,V,NP,NP2,PPS : S1 -> APPS1,NP1,V1,NP3,PPS1
S -> APPS,V,NP,PPS,NP2 : S1 -> APPS1,NP1,PPS1,V1,NP3
S -> APPS,V,NP,PPS,NP2,PPS2 : S1 -> APPS1,NP1,PPS1,V,NP3,PPS3
S -> APPS,V,NP,NP2,EVP : S1 -> APPS1,NP1,V1,NP3
S -> APPS,V,NP,NP2,EVP,PPS : S1 -> APPS1,NP1,V1,NP3,PPS1
S -> APPS,V,NP,PPS,NP2,EVP : S1 -> APPS1,NP1,PPS1,V1,NP3
S -> APPS,V,NP,PPS,NP2,EVP,PPS2 : S1 -> APPS1,NP1,PPS1,V1,NP3,PPS3
S -> APPS,V,NP,NP2,PPS,EVP : S1 -> APPS1,NP1,V1,NP2,PPS1
S -> APPS,V,NP,NP2,PPS,EVP,PPS2 : S1 -> APPS1,NP1,V1,NP3,PPS4
S -> APPS,V,NP,PPS,NP2,PPS2,EVP : S1 -> APPS1,NP1,PPS1,V1,NP3,PPS3
S -> APPS,V,NP,PPS,NP2,PPS2,EVP,PPS4 : S1 -> APPS1,NP1,PPS1,V1,NP3,PPS6

NP -> PN : NP1 -> PN1
NP -> PRON : NP1 -> PRON1
NP -> DET,N : NP1 -> DET1,N1
NP -> DET,ADJP,N : NP1 -> DET1,N1,ADJP1
NP -> DET,POS_PRON : NP1 -> DET1,POS_PRON1

PPS -> PP, PPS : PPS1 -> PP1, PPS1
PPS -> PP : PPS1 -> PP1
PP -> PREP, NP : PP1 -> PREP1, NP1

APPS -> AVD,PPS : APPS ->ADV,PPS
APPS -> AVD : APPS ->ADV
APPS -> PPS : APPS ->PPS

ADJP -> ADJ,ADJP : ADP -> ADJ,ADJP
ADJP -> ADJ : ADJP -> ADJ
```

Donde:

ADJ = Adjetivo
ADJP = Frase Adjetival
ADV = Adverbio
ADVP = Frase Adverbial

DETP = Determinante

EVP = Frase verbal al final de la oración

N = Sustantivo
NP = Frase nominal

PN = Nombre propio
POS_PRON = Pronombre posesivo
PP = Frase Preposicional
PREP = Preposición
PRON = Pronombre
PPS = Frases Preposicionales

S = Oración

V = Verbo finito

WVW = Palabra de inicio de pregunta